BRIEF CONTENTS

CONTENTS

Preface

Data communications and networking may be the fastest growing technologies in our culture today. One of the ramifications of that growth is a dramatic increase in the number of professions where an understanding of these technologies is essential for success- and a proportionate increase in the number and types of students taking courses to learn about them.

Features of the Book

Several features of this text are designed to make it particularly easy for students to understand data communications and networking.

Structure

We have used the five-layer Internet model as the framework for the text not only because a thorough understanding of the model is essential to understanding most current networking theory but also because it is based on a structure of interdependencies: Each layer builds upon the layer beneath it and supports the layer above it. In the same way, each concept introduced in our text builds upon the concepts examined in the previous sections. The Internet model was chosen because it is a protocol that is fully implemented.

This text is designed for students with little or no background in telecommunications or data communications. For this reason, we use a bottom-up approach. With this approach, students learn first about data communications (lower layers) before learning about networking (upper layers).

Visual Approach

The book presents highly technical subject matter without complex formulas by using a balance of text and figures. More than 700 figures accompanying the text provide a visual and intuitive opportunity for understanding the material. Figures are particularly important in explaining networking concepts, which are based on connections and transmission. Both of these ideas are easy to grasp visually.

Highlighted Points

We emphasize important concepts in highlighted boxes for quick reference and immediate attention.

Examples and Applications

When appropriate, we have selected examples to reflect true-to-life situations. For example, in Chapter 6 we have shown several cases of telecommunications in current telephone networks.

Recommended Reading

Each chapter includes a list of books and sites that can be used for further reading.

Key Terms

Each chapter includes a list of key terms for the student.

Summary

Each chapter ends with a summary of the material covered in that chapter. The summary provides a brief overview of all the important points in the chapter.

Practice Set

Each chapter includes a practice set designed to reinforce and apply salient concepts. It consists of three parts: review questions, exercises, and research activities (only for appropriate chapters). Review questions are intended to test the student's first-level understanding of the material presented in the chapter. Exercises require deeper understanding of the materiaL Research activities are designed to create motivation for further study.

Appendixes

The appendixes are intended to provide quick reference material or a review of materials needed to understand the concepts discussed in the book.

Glossary and Acronyms

The book contains an extensive glossary and a list of acronyms.

Changes in the Fourth Edition

The Fourth Edition has major changes from the Third Edition, both in the organization and in the contents.

Organization

The following lists the changes in the organization of the book:

1. Chapter 6 now contains multiplexing as well as spreading.
2. Chapter 8 is now totally devoted to switching.
3. The contents of Chapter 12 are moved to Chapter 11.
4. Chapter 17 covers SONET technology.
5. Chapter 19 discusses IP addressing.
6. Chapter 20 is devoted to the Internet Protocol.
7. Chapter 21 discusses three protocols: ARP, ICMP, and IGMP.
8. Chapter 28 is new and devoted to network management in the Internet.
9. The previous Chapters 29 to 31 are now Chapters 30 to 32.

Contents

We have revised the contents of many chapters including the following:

1. The contents of Chapters 1 to 5 are revised and augmented. Examples are added to clarify the contents.
2. The contents of Chapter 10 are revised and augmented to include methods of error detection and correction.
3. Chapter 11 is revised to include a full discussion of several control link protocols.
4. Delivery, forwarding, and routing of datagrams are added to Chapter 22.
5. The new transport protocol, SCTP, is added to Chapter 23.
6. The contents of Chapters 30, 31, and 32 are revised and augmented to include additional discussion about security issues and the Internet.
7. New examples are added to clarify the understanding of concepts.

End Materials

1. A section is added to the end of each chapter listing additional sources for study.
2. The review questions are changed and updated.
3. The multiple-choice questions are moved to the book site to allow students to self-test their knowledge about the contents of the chapter and receive immediate feedback.
4. Exercises are revised and new ones are added to the appropriate chapters.
5. Some chapters contain research activities.

Instructional Materials

Instructional materials for both the student and the teacher are revised and augmented. The solutions to exercises contain both the explanation and answer including full colored figures or tables when needed. The Powerpoint presentations are more comprehensive and include text and figures.

Contents

The book is divided into seven parts. The first part is an overview; the last part concerns network security. The middle five parts are designed to represent the five layers of the Internet model. The following summarizes the contents of each part.

Part One: Overview

The first part gives a general overview of data communications and networking. Chapter 1 covers introductory concepts needed for the rest of the book. Chapter 2 introduces the Internet model.

Part Two: Physical Layer

The second part is a discussion of the physical layer of the Internet model. Chapters 3 to 6 discuss telecommunication aspects of the physical layer. Chapter 7 introduces the transmission media, which, although not part of the physical layer, is controlled by it. Chapter 8 is devoted to switching, which can be used in several layers. Chapter 9 shows how two public networks, telephone and cable TV, can be used for data transfer.

Part Three: Data Link Layer

The third part is devoted to the discussion of the data link layer of the Internet model. Chapter 10 covers error detection and correction. Chapters 11, 12 discuss issues related to data link control. Chapters 13 through 16 deal with LANs. Chapters 17 and] 8 are about WANs. LANs and WANs are examples of networks operating in the first two layers of the Internet model.

Part Four: Network Layer

The fourth part is devoted to the discussion of the network layer of the Internet model. Chapter 19 covers **IP** addresses. Chapters 20 and 21 are devoted to the network layer protocols such as **IP,** ARP, ICMP, and IGMP. Chapter 22 discusses delivery, forwarding, and routing of packets in the Internet.

Part Five: Transport Layer

The fifth part is devoted to the discussion of the transport layer of the Internet model. Chapter 23 gives an overview of the transport layer and discusses the services and duties of this layer. It also introduces three transport-layer protocols: UDP, TCP, and SCTP. Chapter 24 discusses congestion control and quality of service, two issues related to the transport layer and the previous two layers.

Part Six: Application Layer

The sixth part is devoted to the discussion of the application layer of the Internet model. Chapter 25 is about DNS, the application program that is used by other application programs to map application layer addresses to network layer addresses. Chapter 26 to 29 discuss some common applications protocols in the Internet.

Part Seven: Security

The seventh part is a discussion of security. It serves as a prelude to further study in this subject. Chapter 30 briefly discusses cryptography. Chapter 31 introduces security aspects. Chapter 32 shows how different security aspects can be applied to three layers of the Internet model.

Online Learning Center

The McGraw-Hill Online Learning Center contains much additional material. Available at www.mhhe.com/forouzan. As students read through *Data Communications and Networking,* they can go online to take self-grading quizzes. They can also access lecture materials such as PowerPoint slides, and get additional review from animated figures from the book. Selected solutions are also available over the Web. The solutions to odd-numbered problems are provided to students, and instructors can use a password to access the complete set of solutions.

Additionally, McGraw-Hill makes it easy to create a website for your networking course with an exclusive McGraw-Hill product called PageOut. It requires no prior knowledge of HTML, no long hours, and no design skills on your part. Instead, PageOut offers a series of templates. Simply fill them with your course information and

click on one of 16 designs. The process takes under an hour and leaves you with a professionally designed website.

Although PageOut offers "instant" development, the finished website provides powerful features. An interactive course syllabus allows you to post content to coincide with your lectures, so when students visit your PageOut website, your syllabus will direct them to components of Forouzan's Online Learning Center, or specific material of your own.

How to Use the Book

This book is written for both an academic and a professional audience. The book can be used as a self-study guide for interested professionals. As a textbook, it can be used for a one-semester or one-quarter course. The following are some guidelines.

- Parts one to three are strongly recommended.
- Parts four to six can be covered if there is no following course in *TCP/IP* protocol.
- Part seven is recommended if there is no following course in network security.

Acknowledgments

It is obvious that the development of a book of this scope needs the support of many people.

Peer Review

The most important contribution to the development of a book such as this comes from peer reviews. We cannot express our gratitude in words to the many reviewers who spent numerous hours reading the manuscript and providing us with helpful comments and ideas. We would especially like to acknowledge the contributions of the following reviewers for the third and fourth editions of this book.

Farid Ahmed, *Catholic University*
Kaveh Ashenayi, *University of Tulsa*
Yoris Au, *University of Texas, San Antonio*
Essie Bakhtiar, *Clayton College & State University*
Anthony Barnard, *University of Alabama, Brimingham*
A.T. Burrell, *Oklahoma State University*
Scott Campbell, *Miami University*
Teresa Carrigan, *Blackburn College*
Hwa Chang, *Tufts University*
Edward Chlebus, *Illinois Institute of Technology*
Peter Cooper, *Sam Houston State University*
Richard Coppins, *Virginia Commonwealth University*
Harpal Dhillon, *Southwestern Oklahoma State University*
Hans-Peter Dommel, *Santa Clara University*
M. Barry Dumas, *Baruch College, CUNY*
William Figg, *Dakota State University*
Dale Fox, *Quinnipiac University*
Terrence Fries, *Coastal Carolina University*
Errin Fulp, *Wake Forest University*

Sandeep Gupta, *Arizona State University*
George Hamer, *South Dakota State University*
James Henson, *California State University, Fresno*
Tom Hilton, *Utah State University*
Allen Holliday, *California State University, Fullerton*
Seyed Hossein Hosseini, *University of Wisconsin, Milwaukee*
Gerald Isaacs, *Carroll College, Waukesha*
Hrishikesh Joshi, *DeVry University*
E.S. Khosravi, *Southern University*
Bob Kinicki, *Worcester Polytechnic University*
Kevin Kwiat, *Hamilton College*
Ten-Hwang Lai, *Ohio State University*
Chung-Wei Lee, *Auburn University*
Ka-Cheong Leung, *Texas Tech University*
Gertrude Levine, *Fairleigh Dickinson University*
Alvin Sek See Lim, *Auburn University*
Charles Liu, *California State University, Los Angeles*
Wenhang Liu, *California State University, Los Angeles*
Mark Llewellyn, *University of Central Florida*
Sanchita Mal-Sarkar, *Cleveland State University*
Louis Marseille, *Harford Community College*
Kevin McNeill, *University of Arizona*
Arnold C. Meltzer, *George Washington University*
Rayman Meservy, *Brigham Young University*
Prasant Mohapatra, *University of California, Davis*
Hung Z Ngo, *SUNY, Buffalo*
Larry Owens, *California State University, Fresno*
Arnold Patton, *Bradley University*
Dolly Samson, *Hawaii Pacific University*
Joseph Sherif, *California State University, Fullerton*
Robert Simon, *George Mason University*
Ronald 1. Srodawa, *Oakland University*
Daniel Tian, *California State University, Monterey Bay*
Richard Tibbs, *Radford University*
Christophe Veltsos, *Minnesota State University, Mankato*
Yang Wang, *University of Maryland, College Park*
Sherali Zeadally, *Wayne State University*

McGraw-Hill Staff

Special thanks go to the staff of McGraw-Hill. Alan Apt, our publisher, proved how a proficient publisher can make the impossible possible. Rebecca Olson, the developmental editor, gave us help whenever we needed it. Sheila Frank, our project manager, guided us through the production process with enormous enthusiasm. We also thank David Hash in design, Kara Kudronowicz in production, and Patti Scott, the copy editor.

PART 1

Overview

Objectives

Part 1 provides a general idea of what we will see in the rest of the book. Four major concepts are discussed: data communications, networking, protocols and standards, and networking models.

Networks exist so that data may be sent from one place to another-the basic concept of *data communications.* To fully grasp this subject, we must understand the data communication components, how different types of data can be represented, and how to create a data flow.

Data communications between remote parties can be achieved through a process called *networking,* involving the connection of computers, media, and networking devices. Networks are divided into two main categories: local area networks (LANs) and wide area networks (WANs). These two types of networks have different characteristics and different functionalities. The Internet, the main focus of the book, is a collection of LANs and WANs held together by internetworking devices.

Protocols and standards are vital to the implementation of data communications and networking. Protocols refer to the rules; a standard is a protocol that has been adopted by vendors and manufacturers.

Network models serve to organize, unify, and control the hardware and software components of data communications and networking. Although the term "network model" suggests a relationship to networking, the model also encompasses data communications.

Chapters

This part consists of two chapters: Chapter 1 and Chapter 2.

Chapter 1

In Chapter 1, we introduce the concepts of data communications and networking. We discuss data communications components, data representation, and data flow. We then move to the structure of networks that carry data. We discuss network topologies, categories of networks, and the general idea behind the Internet. The section on protocols and standards gives a quick overview of the organizations that set standards in data communications and networking.

Chapter 2

The two dominant networking models are the Open Systems Interconnection (OSI) and the Internet model (TCP/IP).The first is a theoretical framework; the second is the actual model used in today's data communications. **In** Chapter 2, we first discuss the OSI model to give a general background. We then concentrate on the Internet model, which is the foundation for the rest of the book.

CHAPTERl

Introduction

Data communications and networking are changing the way we do business and the way we live. Business decisions have to be made ever more quickly, and the decision makers require immediate access to accurate information. Why wait a week for that report from Germany to arrive by mail when it could appear almost instantaneously through computer networks? Businesses today rely on computer networks and internetworks. But before we ask how quickly we can get hooked up, we need to know how networks operate, what types of technologies are available, and which design best fills which set of needs.

The development of the personal computer brought about tremendous changes for business, industry, science, and education. A similar revolution is occurring in data communications and networking. Technological advances are making it possible for communications links to carry more and faster signals. As a result, services are evolving to allow use of this expanded capacity. For example, established telephone services such as conference calling, call waiting, voice mail, and caller **ID** have been extended.

Research in data communications and networking has resulted in new technologies. One goal is to be able to exchange data such as text, audio, and video from all points in the world. We want to access the Internet to download and upload information quickly and accurately and at any time.

This chapter addresses four issues: data communications, networks, the Internet, and protocols and standards. First we give a broad definition of data communications. Then we define networks as a highway on which data can travel. The Internet is discussed as a good example of an internetwork (i.e., a network of networks). Finally, we discuss different types of protocols, the difference between protocols and standards, and the organizations that set those standards.

1.1 DATA COMMUNICATIONS

When we communicate, we are sharing information. This sharing can be local or remote. Between individuals, local communication usually occurs face to face, while remote communication takes place over distance. The term *telecommunication,* which

includes telephony, telegraphy, and television, means communication at a distance *(tele* is Greek for "far").

The word *data* refers to information presented in whatever form is agreed upon by the parties creating and using the data.

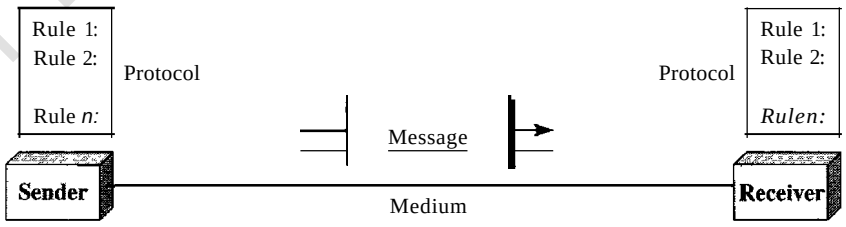
Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable. For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs). The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. Delivery. The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.
2. Accuracy. The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.
3. Timeliness. The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called *real-time* transmission.
4. Jitter. Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 3D ms. If some of the packets arrive with 3D-ms delay and others with 4D-ms delay, an uneven quality in the video is the result.

COinponents

A data communications system has five components (see Figure 1.1).

Figure 1.1 *Five components ofdata communication*



1. Message. The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. Sender. The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
3. Receiver. The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. Transmission medium. The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.

5. Protocol. A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

Data Representation

Information today comes in different forms such as text, numbers, images, audio, and video.

Text

In data communications, text is represented as a bit pattern, a sequence of bits (Os or Is). Different sets of bit patterns have been designed to represent text symbols. Each set is called a code, and the process of representing symbols is called coding. Today, the prevalent coding system is called Unicode, which uses 32 bits to represent a symbol or character used in any language in the world. The American Standard Code for Information Interchange (ASCII), developed some decades ago in the United States, now constitutes the first 127 characters in Unicode and is also referred to as Basic Latin. Appendix A includes part of the Unicode.

Numbers

Numbers are also represented by bit patterns. However, a code such as ASCII is not used to represent numbers; the number is directly converted to a binary number to simplify mathematical operations. Appendix B discusses several different numbering systems.

Images

Images are also represented by bit patterns. In its simplest form, an image is composed of a matrix of pixels (picture elements), where each pixel is a small dot. The size of the pixel depends on the *resolution.* For example, an image can be divided into 1000 pixels or 10,000 pixels. In the second case, there is a better representation of the image (better resolution), but more memory is needed to store the image.

After an image is divided into pixels, each pixel is assigned a bit pattern. The size and the value of the pattern depend on the image. For an image made of only black-and-white dots (e.g., a chessboard), a I-bit pattern is enough to represent a pixel.

If an image is not made of pure white and pure black pixels, you can increase the size of the bit pattern to include gray scale. For example, to show four levels of gray scale, you can use 2-bit patterns. A black pixel can be represented by 00, a dark gray pixel by 01, a light gray pixel by 10, and a white pixel by 11.

There are several methods to represent color images. One method is called RGB, so called because each color is made of a combination of three primary colors: *red,* green, and blue. The intensity of each color is measured, and a bit pattern is assigned to it. Another method is called YCM, in which a color is made of a combination of three other primary colors: yellow, cyan, and magenta.

Audio

Audio refers to the recording or broadcasting of sound or music. Audio is by nature different from text, numbers, or images. It is continuous, not discrete. Even when we

use a microphone to change voice or music to an electric signal, we create a continuous signal. In Chapters 4 and 5, we learn how to change sound or music to a digital or an analog signal.

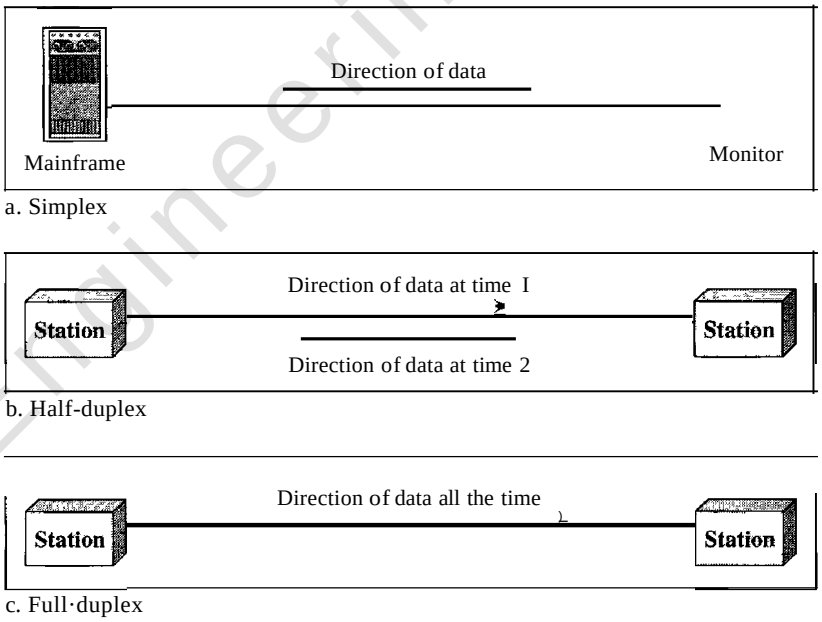
Video

Video refers to the recording or broadcasting of a picture or movie. Video can either be produced as a continuous entity (e.g., by a TV camera), or it can be a combination of images, each a discrete entity, arranged to convey the idea of motion. Again we can change video to a digital or an analog signal, as we will see in Chapters 4 and 5.

Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex as shown in Figure 1.2.

Figure 1.2 *Data flow (simplex, half-duplex, and full-duplex)*



Simplex

In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive (see Figure 1.2a).

Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output. The simplex mode can use the entire capacity of the channel to send data in one direction.

Half-Duplex

In half-duplex mode, each station can both transmit and receive, but not at the same time. : When one device is sending, the other can only receive, and vice versa (see Figure 1.2b).

The half-duplex mode is like a one-lane road with traffic allowed in both directions. When cars are traveling in one direction, cars going the other way must wait. In a half-duplex transmission, the entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time. Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

The half-duplex mode is used in cases where there is no need for communication in both directions at the same time; the entire capacity of the channel can be utilized for each direction.

Full-Duplex

In full-duplex **mode** (also called duplex), both stations can transmit and receive simultaneously (see Figure 1.2c).

The full-duplex mode is like a two-way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link: with signals going in the other direction. This sharing can occur in two ways: Either the link must contain two physically separate transmission paths, one for sending and the other for receiving; or the capacity of the channel is divided between signals traveling in both directions.

One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time.

The full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.

1.2 NETWORKS

A network is a set of devices (often referred to as *nodes)* connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

Distributed Processing

Most networks use distributed processing, in which a task is divided among multiple computers. Instead of one single large machine being responsible for all aspects of a process, separate computers (usually a personal computer or workstation) handle a subset.

Network Criteria

A network must be able to meet a certain number of criteria. The most important of these are performance, reliability, and security.

Performance

Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to

another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

Performance is often evaluated by two networking metrics: throughput and delay. We often need more throughput and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

Reliability

In addition to accuracy of delivery, network reliability is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

Security

Network security issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

Physical Structures

Before discussing networks, we need to define some network attributes.

Type of Connection

A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another. For visualization purposes, it is simplest to imagine any link as a line drawn between two points. For communication to occur, two devices must be connected in some way to the same link at the same time. There are two possible types of connections: point-to-point and multipoint.

Point-to-Point A point-to-point connection provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible (see Figure 1.3a). When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

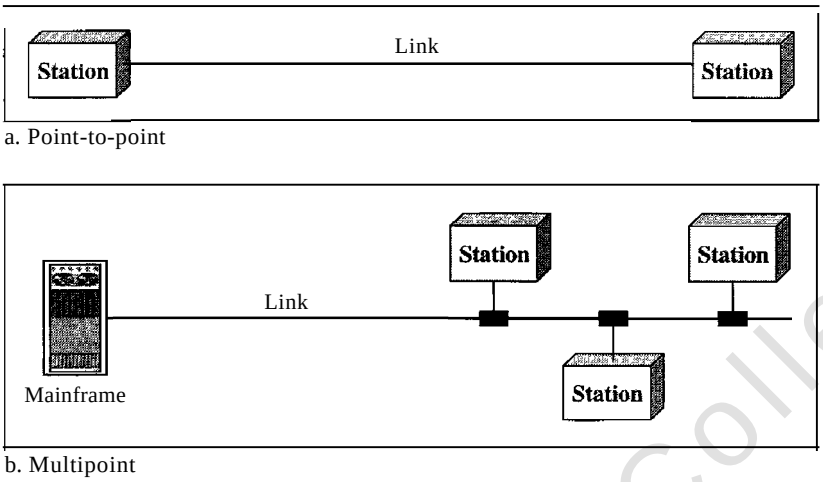
Multipoint A multipoint (also called multidrop) connection is one in which more than two specific devices share a single link (see Figure 1.3b).

In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a *spatially shared* connection. If users must take turns, it is a *timeshared* connection.

Physical Topology

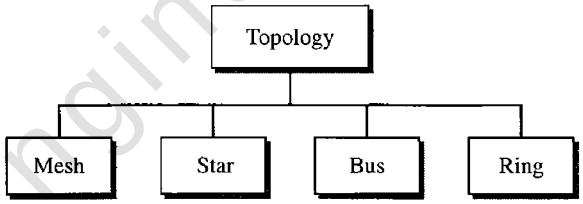
The term *physical topology* refers to the way in which a network is laid out physically.: Two or more devices connect to a link; two or more links form a topology. The topology

Figure 1.3 *Types of connections: point-to-point and multipoint*



of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. There are four basic topologies possible: mesh, star, bus, and ring (see Figure 1.4).

Figure **1.4** *Categories of topology*



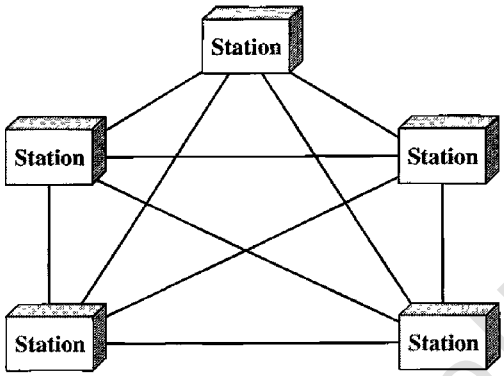
Mesh In a mesh topology, every device has a dedicated point-to-point link to every other device. The term *dedicated* means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with n nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to $n - 1$ nodes, node 2 must be connected to $n - 1$ nodes, and finally node n must be connected to $n - 1$ nodes. We need $n(n - 1)$ physical links. However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need

$$n(n-1)/2$$

duplex-mode links.

To accommodate that many links, every device on the network must have $n - 1$ input/output *(VO)* ports (see Figure 1.5) to be connected to the other $n - 1$ stations.

Figure 1.5 *A fully connected mesh topology (five devices)*



A mesh offers several advantages over other network topologies. First, the use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices. Second, a mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system. Third, there is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages. Finally, point-to-point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate. Finally, the hardware required to connect each link (I/O ports and cable) can be prohibitively expensive. For these reasons a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies.

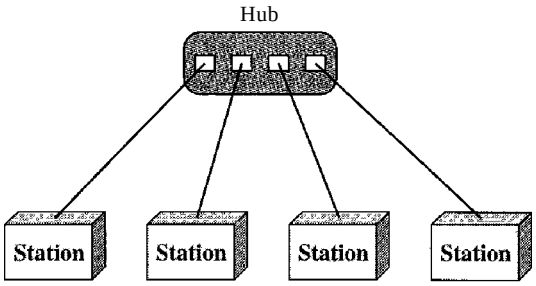
One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

Star Topology In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a hub. The devices are not directly linked to one another. Unlike a mesh topology, a star topology does not allow direct traffic between devices. The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device (see Figure 1.6) .

A star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect it to any number of others. This factor also makes it easy to install and reconfigure. Far less cabling needs to be housed, and additions, moves, and deletions involve only one connection: between that device and the hub.

Other advantages include robustness. If one link fails, only that link is affected. All other links remain active. This factor also lends itself to easy fault identification and

Figure 1.6 *A star topology connecting four stations*



fault isolation. As long as the hub is working, it can be used to monitor link problems and bypass defective links.

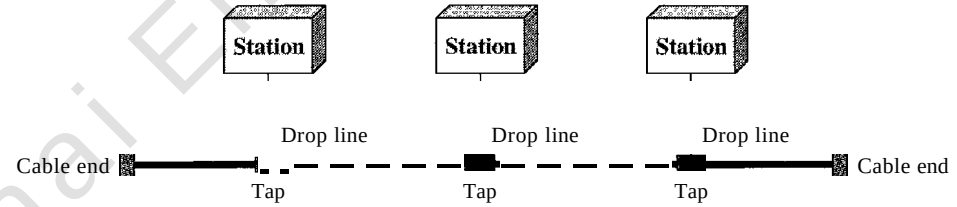
One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead.

Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is required in a star than in some other topologies (such as ring or bus).

The star topology is used in local-area networks (LANs), as we will see in Chapter 13. High-speed LANs often use a star topology with a central hub.

Bus Topology The preceding examples all describe point-to-point connections. A **bus topology,** on the other hand, is multipoint. One long cable acts as a **backbone** to link all the devices in a network (see Figure 1.7).

Figure 1.7 *A bus topology connecting three stations*



Nodes are connected to the bus cable by drop lines and taps. A drop line is a connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching

all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the backbone.

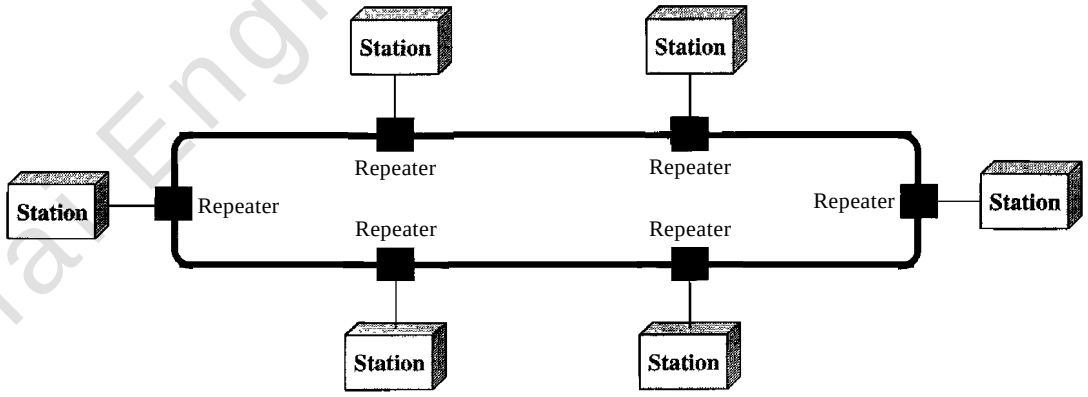
Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected to a given length of cable. Adding new devices may therefore require modification or replacement of the backbone.

In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

Bus topology was the one of the first topologies used in the design of early local-area networks. Ethernet LANs can use a bus topology, but they are less popular now for reasons we will discuss in Chapter 13.

Ring Topology In a ring topology, each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along (see Figure 1.8).

Figure 1.8 *A ring topology connecting six stations*



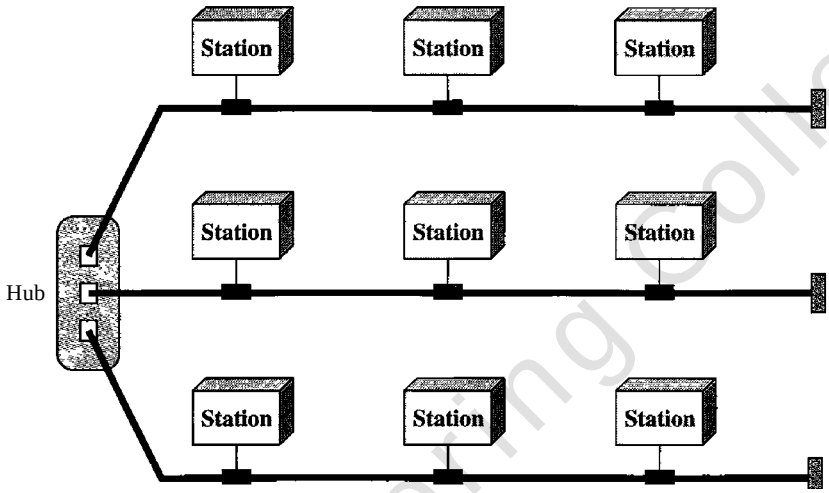
A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally in a ring, a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

Ring topology was prevalent when IBM introduced its local-area network Token Ring. Today, the need for higher-speed LANs has made this topology less popular.

Hybrid Topology A network can be hybrid. For example, we can have a main star topology with each branch connecting several stations in a bus topology as shown in Figure 1.9.

Figure 1.9 A *hybrid topology: a star backbone with three bus networks*



Network Models

Computer networks are created by different entities. Standards are needed so that these heterogeneous networks can communicate with one another. The two best-known standards are the OSI model and the Internet model. In Chapter 2 we discuss these two models. The OSI (Open Systems Interconnection) model defines a seven-layer network; the Internet model defines a five-layer network. This book is based on the Internet model with occasional references to the OSI model.

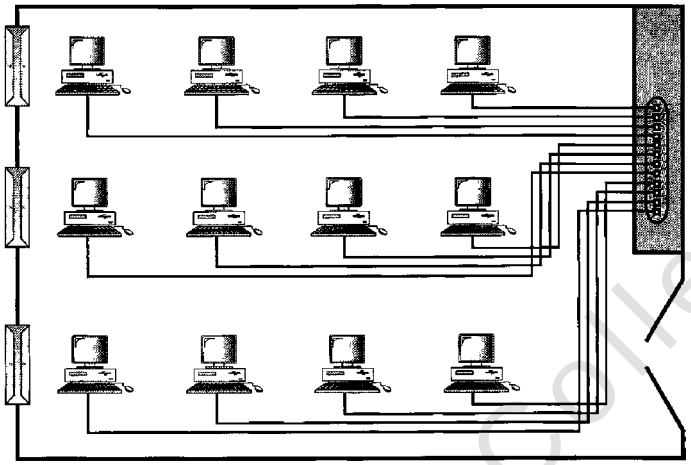
Categories of Networks

Today when we speak of networks, we are generally referring to two primary categories: local-area networks and wide-area networks. The category into which a network falls is determined by its size. A LAN normally covers an area less than 2 mi; a WAN can be worldwide. Networks of a size in between are normally referred to as metropolitan-area networks and span tens of miles.

Local Area Network

A local area network (LAN) is usually privately owned and links the devices in a single office, building, or campus (see Figure 1.10). Depending on the needs of an organization and the type of technology used, a LAN can be as simple as two PCs and a printer in someone's home office; or it can extend throughout a company and include audio and video peripherals. Currently, LAN size is limited to a few kilometers.

Figure 1.10 *An isolated IAN connecting* 12 *computers to a hub in a closet*



LANs are designed to allow resources to be shared between personal computers or workstations. The resources to be shared can include hardware (e.g., a printer), software (e.g., an application program), or data. A common example of a LAN, found in many business environments, links a workgroup of task-related computers, for example, engineering workstations or accounting PCs. One of the computers may be given a large-capacity disk drive and may become a server to clients. Software can be stored on this central server and used as needed by the whole group. In this example, the size of the LAN may be determined by licensing restrictions on the number of users per copy of software, or by restrictions on the number of users licensed to access the operating system.

In addition to size, LANs are distinguished from other types of networks by their transmission media and topology. In general, a given LAN will use only one type of transmission medium. The most common LAN topologies are bus, ring, and star.

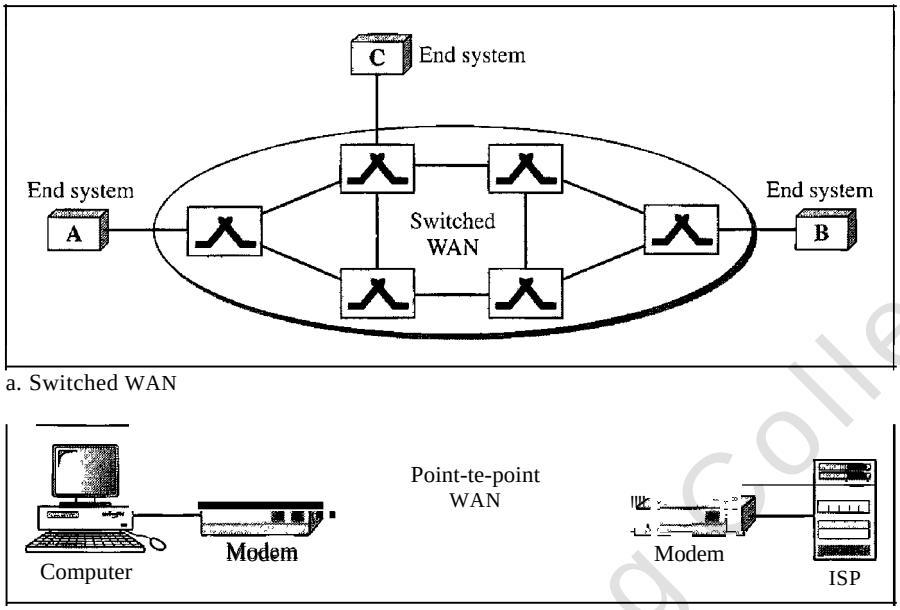
Early LANs had data rates in the 4 to 16 megabits per second (Mbps) range. Today, however, speeds are normally 100 or 1000 Mbps. LANs are discussed at length in Chapters 13, 14, and 15.

Wireless LANs are the newest evolution in LAN technology. We discuss wireless LANs in detail in Chapter 14.

Wide Area Network

A wide area network (WAN) provides long-distance transmission of data, image, audio, and video information over large geographic areas that may comprise a country, a continent, or even the whole world. In Chapters 17 and 18 we discuss wide-area networks in greater detail. A WAN can be as complex as the backbones that connect the Internet or as simple as a dial-up line that connects a home computer to the Internet. We normally refer to the first as a switched WAN and to the second as a point-to-point WAN (Figure 1.11). The switched WAN connects the end systems, which usually comprise a router (internetworking connecting device) that connects to another LAN or WAN. The point-to-point WAN is normally a line leased from a telephone or cable TV provider that connects a home computer or a small LAN to an Internet service provider (lSP). This type of WAN is often used to provide Internet access.

Figure 1.11 *WANs: a switched WAN and a point-to-point WAN*



An early example of a switched WAN is X.25, a network designed to provide connectivity between end users. As we will see in Chapter 18, X.25 is being gradually replaced by a high-speed, more efficient network called Frame Relay. A good example of a switched WAN is the asynchronous transfer mode (ATM) network, which is a network with fixed-size data unit packets called cells. We will discuss ATM in Chapter 18. Another example ofWANs is the wireless WAN that is becoming more and more popular. We discuss wireless WANs and their evolution in Chapter 16.

Metropolitan Area Networks

A metropolitan area network (MAN) is a network with a size between a LAN and a WAN. It normally covers the area inside a town or a city. It is designed for customers who need a high-speed connectivity, normally to the Internet, and have endpoints spread over a city or part of city. A good example of a MAN is the part of the telephone company network that can provide a high-speed DSL line to the customer. Another example is the cable TV network that originally was designed for cable TV, but today can also be used for high-speed data connection to the Internet. We discuss DSL lines and cable TV networks in Chapter 9.

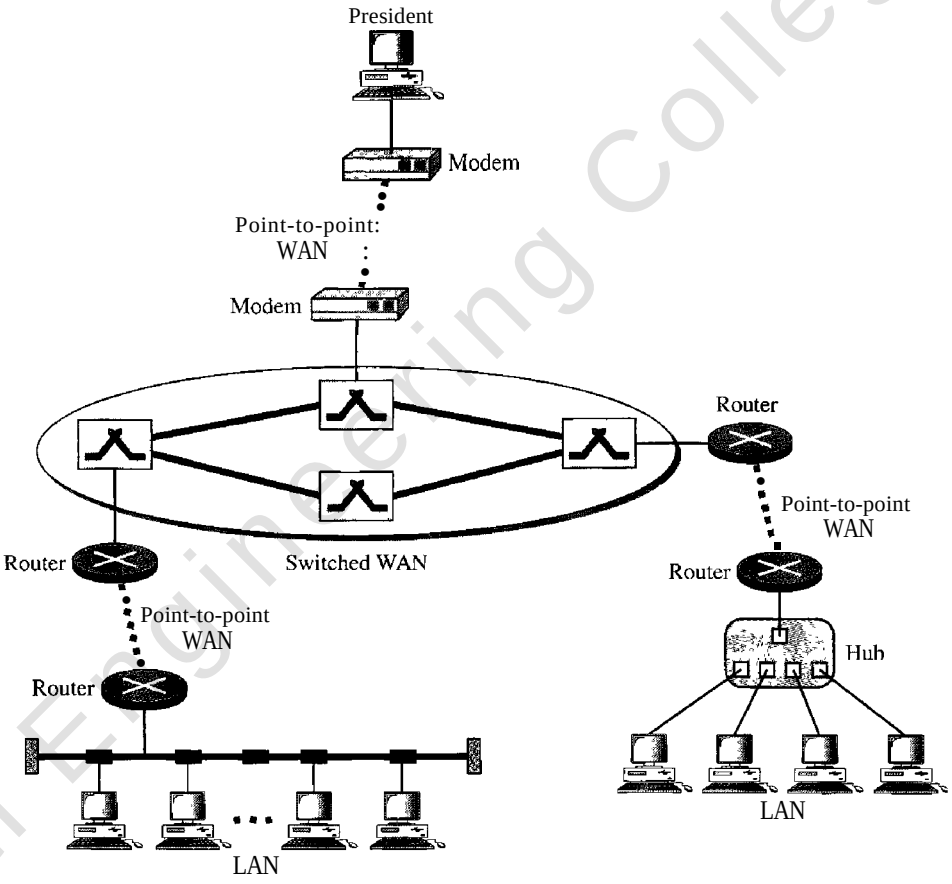
Interconnection of Networks: Internetwork

Today, it is very rare to see a LAN, a MAN, or a LAN in isolation; they are connected to one another. When two or more networks are connected, they become an internetwork, or internet.

As an example, assume that an organization has two offices, one on the east coast and the other on the west coast. The established office on the west coast has a bus topology LAN; the newly opened office on the east coast has a star topology LAN. The president of the company lives somewhere in the middle and needs to have control over the company

from her horne. To create a backbone WAN for connecting these three entities (two LANs and the president's computer), a switched WAN (operated by a service provider such as a telecom company) has been leased. To connect the LANs to this switched WAN, however, three point-to-point WANs are required. These point-to-point WANs can be a high-speed DSL line offered by a telephone company or a cable modern line offered by a cable TV provider as shown in Figure 1.12.

Figure 1.12 *A heterogeneous network made offour WANs and two LANs*



1.3 THE INTERNET

The Internet has revolutionized many aspects of our daily lives. It has affected the way we do business as well as the way we spend our leisure time. Count the ways you've used the Internet recently. Perhaps you've sent electronic mail (e-mail) to a business associate, paid a utility bill, read a newspaper from a distant city, or looked up a local movie schedule-all by using the Internet. Or maybe you researched a medical topic, booked a hotel reservation, chatted with a fellow Trekkie, or comparison-shopped for a car. The Internet is a communication system that has brought a wealth of information to our fingertips and organized it for our use.

The Internet is a structured, organized system. We begin with a brief history of the Internet. We follow with a description of the Internet today.

A Brief History

A network is a group of connected communicating devices such as computers and printers. An internet (note the lowercase letter i) is two or more networks that can communicate with each other. The most notable internet is called the Internet (uppercase letter I), a collaboration of more than hundreds of thousands of interconnected networks. Private individuals as well as various organizations such as government agencies, schools, research facilities, corporations, and libraries in more than 100 countries use the Internet. Millions of people are users. Yet this extraordinary communication system only came into being in 1969.

In the mid-1960s, mainframe computers in research organizations were standalone devices. Computers from different manufacturers were unable to communicate with one another. The Advanced Research Projects Agency (ARPA) in the Department of Defense (DoD) was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for ARPANET, a small network of connected computers. The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an *interface message processor* (IMP). The IMPs, in tum, would be connected to one another. Each IMP had to be able to communicate with other IMPs as well as with its own attached host.

By 1969, ARPANET was a reality. Four nodes, at the University of California at Los Angeles (UCLA), the University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI), and the University of Utah, were connected via the IMPs to form a network. Software called the *Network Control Protocol* (NCP) provided communication between the hosts.

In 1972, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the *Internetting Project.* Cerf and Kahn's landmark 1973 paper outlined the protocols to achieve end-to-end delivery of packets. This paper on Transmission Control Protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway.

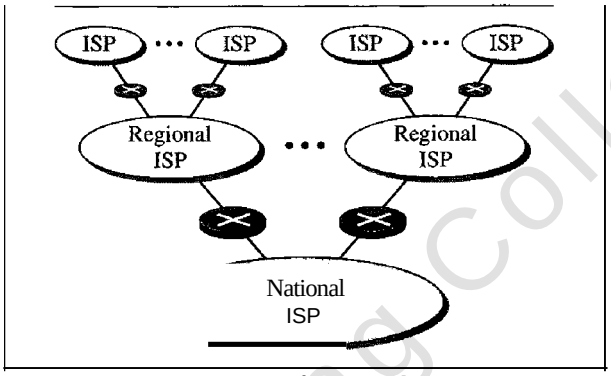
Shortly thereafter, authorities made a decision to split TCP into two protocols: Transmission Control Protocol (TCP) and Internetworking Protocol (lP). IP would handle datagram routing while TCP would be responsible for higher-level functions such as segmentation, reassembly, and error detection. The internetworking protocol became known as TCPIIP.

The Internet Today

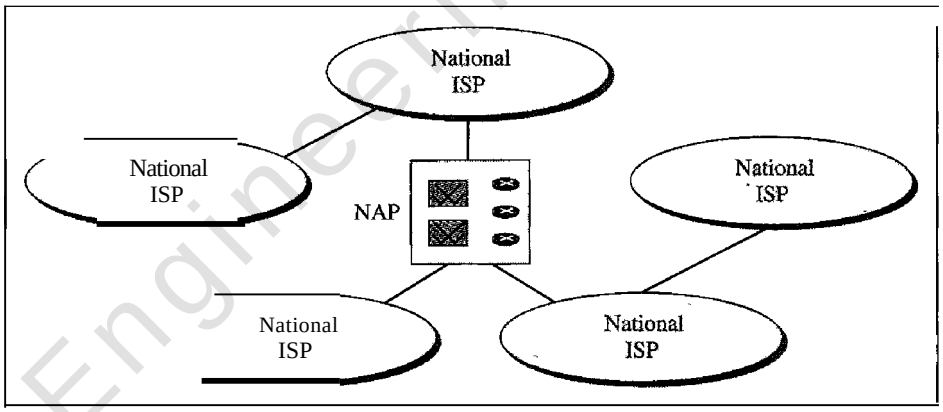
The Internet has come a long way since the 1960s. The Internet today is not a simple hierarchical structure. It is made up of many wide- and local-area networks joined by connecting devices and switching stations. It is difficult to give an accurate representation of the Internet because it is continually changing-new networks are being added, existing networks are adding addresses, and networks of defunct companies are being removed. Today most end users who want Internet connection use the services of Internet service providers (lSPs). There are international service providers, national

service providers, regional service providers, and local service providers. The Internet today is run by private companies, not the government. Figure 1.13 shows a conceptual (not geographic) view of the Internet.

Figure 1.13 *Hierarchical organization of the Internet*



a. Structure of a national ISP



b. Interconnection of national ISPs

International Internet Service Providers

At the top of the hierarchy are the international service providers that connect nations together.

National Internet Service Providers

The national Internet service providers are backbone networks created and maintained by specialized companies. There are many national ISPs operating in North America; some of the most well known are SprintLink, PSINet, UUNet Technology, AGIS, and internet Mel. To provide connectivity between the end users, these backbone networks are connected by complex switching stations (normally run by a third party) called network access points (NAPs). Some national ISP networks are also connected to one another by private switching stations called *peering points.* These normally operate at a high data rate (up to 600 Mbps).

Regional Internet Service Providers

Regional internet service providers or regional ISPs are smaller ISPs that are connected to one or more national ISPs. They are at the third level of the hierarchy with a smaller data rate.

Local Internet Service Providers

Local Internet service providers provide direct service to the end users. The local ISPs can be connected to regional ISPs or directly to national ISPs. Most end users are connected to the local ISPs. Note that in this sense, a local ISP can be a company that just provides Internet services, a corporation with a network that supplies services to its own employees, or a nonprofit organization, such as a college or a university, that runs its own network. Each of these local ISPs can be connected to a regional or national service provider.

1.4 PROTOCOLS AND STANDARDS

In this section, we define two widely used terms: protocols and standards. First, we define *protocol*, which is synonymous with *rule*. Then we discuss *standards*, which are agreed-upon rules.

Protocols

In computer networks, communication occurs between entities in different systems. An entity is anything capable of sending or receiving information. However, two entities cannot simply send bit streams to each other and expect to be understood. For communication to occur, the entities must agree on a protocol. A protocol is a set of rules that govern data communications. A protocol defines what is communicated, how it is communicated, and when it is communicated. The key elements of a protocol are syntax, semantics, and timing.

- Syntax. The term *syntax* refers to the structure or format of the data, meaning the order in which they are presented. For example, a simple protocol might expect the first 8 bits of data to be the address of the sender, the second 8 bits to be the address of the receiver, and the rest of the stream to be the message itself.
- Semantics. The word *semantics* refers to the meaning of each section of bits. How is a particular pattern to be interpreted, and what action is to be taken based on that interpretation? For example, does an address identify the route to be taken or the final destination of the message?
- Timing. The term *timing* refers to two characteristics: when data should be sent and how fast they can be sent. For example, if a sender produces data at 100 Mbps but the receiver can process data at only 1 Mbps, the transmission will overload the receiver and some data will be lost.

Standards

Standards are essential in creating and maintaining an open and competitive market for equipment manufacturers and in guaranteeing national and international interoperability of data and telecommunications technology and processes. Standards provide guidelines

to manufacturers, vendors, government agencies, and other service providers to ensure the kind of interconnectivity necessary in today's marketplace and in international communications. Data communication standards fall into two categories: *de facto* (meaning "by fact" or "by convention") and *de jure* (meaning "by law" or "by regulation").

- De facto. Standards that have not been approved by an organized body but have been adopted as standards through widespread use are de facto standards. De facto standards are often established originally by manufacturers who seek to define the functionality of a new product or technology.
- De jure. Those standards that have been legislated by an officially recognized body are de jure standards.

Standards Organizations

Standards are developed through the cooperation of standards creation committees, forums, and government regulatory agencies.

Standards Creation Committees

While many organizations are dedicated to the establishment of standards, data telecommunications in North America rely primarily on those published by the following:

- International Organization for Standardization (ISO). The ISO is a multinational body whose membership is drawn mainly from the standards creation committees of various governments throughout the world. The ISO is active in developing cooperation in the realms of scientific, technological, and economic activity.
- International Telecommunication Union-Telecommunication Standards Sector (ITU-T). By the early 1970s, a number of countries were defining national standards for telecommunications, but there was still little international compatibility. The United Nations responded by forming, as part of its International Telecommunication Union (ITU), a committee, the Consultative Committee for International Telegraphy and Telephony (CCITT). This committee was devoted to the research and establishment of standards for telecommunications in general and for phone and data systems in particular. On March 1, 1993, the name of this committee was changed to the International Telecommunication Union-Telecommunication Standards Sector (ITU-T).
- American National Standards Institute (ANSI). Despite its name, the American National Standards Institute is a completely private, nonprofit corporation not affiliated with the U.S. federal government. However, all ANSI activities are undertaken with the welfare of the United States and its citizens occupying primary importance.
- Institute of Electrical and Electronics Engineers (IEEE). The Institute of Electrical and Electronics Engineers is the largest professional engineering society in the world. International in scope, it aims to advance theory, creativity, and product quality in the fields of electrical engineering, electronics, and radio as well as in all related branches of engineering. As one of its goals, the IEEE oversees the development and adoption of international standards for computing and communications.
- Electronic Industries Association (EIA). Aligned with ANSI, the Electronic Industries Association is a nonprofit organization devoted to the promotion of

electronics manufacturing concerns. Its activities include public awareness education and lobbying efforts in addition to standards development. In the field of information technology, the EIA has made significant contributions by defining physical connection interfaces and electronic signaling specifications for data communication.

Forums

Telecommunications technology development is moving faster than the ability of standards committees to ratify standards. Standards committees are procedural bodies and by nature slow-moving. To accommodate the need for working models and agreements and to facilitate the standardization process, many special-interest groups have developed **forums** made up of representatives from interested corporations. The forums work with universities and users to test, evaluate, and standardize new technologies. By concentrating their efforts on a particular technology, the forums are able to speed acceptance and use of those technologies in the telecommunications community. The forums present their conclusions to the standards bodies.

Regulatory Agencies

All communications technology is subject to regulation by government agencies such as the **Federal Communications Commission** (FCC) in the United States. The purpose of these agencies is to protect the public interest by regulating radio, television, and wire/cable communications. The FCC has authority over interstate and international commerce as it relates to communications.

Internet Standards

An **Internet standard** is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status. A specification begins as an Internet draft. An **Internet draft** is a working document (a work in progress) with no official status and a 6-month lifetime. Upon recommendation from the Internet authorities, a draft may be published as a **Request for Comment** (RFC). Each RFC is edited, assigned a number, and made available to all interested parties. RFCs go through maturity levels and are categorized according to their requirement level.

1.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items enclosed in brackets [...] refer to the reference list at the end of the book.

Books

The introductory materials covered in this chapter can be found in [Sta04] and [PD03]. [Tan03] discusses standardization in Section 1.6.

Sites

The following sites are related to topics discussed in this chapter.

- www.acm.org/sigcomm/sos.html This site gives the status of various networking standards.
- www.ietf.org/ The Internet Engineering Task Force (IETF) home page.

RFCs

The following site lists all RFCs, including those related to IP and TCP. In future chapters we cite the RFCs pertinent to the chapter material.

- www.ietf.org/rfc.html

1.6 KEY TERMS

Advanced Research Projects Agency (ARPA)
American National Standards Institute (ANSI)
American Standard Code for Information Interchange (ASCII)
ARPANET
audio
backbone
Basic Latin
bus topology
code
Consultative Committee for International Telegraphy and Telephony (CCITT)
data
data communications
de facto standards
de jure standards
delay
distributed processing
Electronic Industries Association (EIA)
entity
Federal Communications Commission (FCC)
forum
full-duplex mode, or duplex
half-duplex mode
hub
image
Institute of Electrical and Electronics Engineers (IEEE)
International Organization for Standardization (ISO)
International Telecommunication Union-Telecommunication Standards Sector (ITU-T)
Internet
Internet draft
Internet service provider (ISP)
Internet standard
internetwork or internet
local area network (LAN)
local Internet service providers
mesh topology
message
metropolitan area network (MAN)
multipoint or multidrop connection
national Internet service provider
network

network access points (NAPs)
node
performance
physical topology
point-to-point connection
protocol
receiver
regional ISP
reliability
Request for Comment (RFC)
ROB
ring topology
security
semantics
sender
simplex mode
star topology
syntax
telecommunication
throughput
timing
Transmission Control Protocol! Internetworking Protocol (TCPIIP)
transmission medium
Unicode
video
wide area network (WAN)
YCM

1.7 SUMMARY

- Data communications are the transfer of data from one device to another via some form of transmission medium.
- A data communications system must transmit data to the correct destination in an accurate and timely manner.
- The five components that make up a data communications system are the message, sender, receiver, medium, and protocol.
- Text, numbers, images, audio, and video are different forms of information.
- Data flow between two devices can occur in one of three ways: simplex, half-duplex, or full-duplex.
- A network is a set of communication devices connected by media links.
- In a point-to-point connection, two and only two devices are connected by a dedicated link. In a multipoint connection, three or more devices share a link.
- Topology refers to the physical or logical arrangement of a network. Devices may be arranged in a mesh, star, bus, or ring topology.
- A network can be categorized as a local area network or a wide area network.
- A LAN is a data communication system within a building, plant, or campus, or between nearby buildings.
- A WAN is a data communication system spanning states, countries, or the whole world.
- An internet is a network of networks.
- The Internet is a collection of many separate networks.
- There are local, regional, national, and international Internet service providers.
- A protocol is a set of rules that govern data communication; the key elements of a protocol are syntax, semantics, and timing.

- Standards are necessary to ensure that products from different manufacturers can work together as expected.
- The ISO, ITD-T, ANSI, IEEE, and EIA are some of the organizations involved in standards creation.
- Forums are special-interest groups that quickly evaluate and standardize new technologies.
- A Request for Comment is an idea or concept that is a precursor to an Internet standard.

1.8 PRACTICE SET

Review Questions

1. Identify the five components of a data communications system.
2. What are the advantages of distributed processing?
3. What are the three criteria necessary for an effective and efficient network?
4. What are the advantages of a multipoint connection over a point-to-point connection?
5. What are the two types of line configuration?
6. Categorize the four basic topologies in terms of line configuration.
7. What is the difference between half-duplex and full-duplex transmission modes?
8. Name the four basic network topologies, and cite an advantage of each type.
9. For *n* devices in a network, what is the number of cable links required for a mesh, ring, bus, and star topology?
10. What are some of the factors that determine whether a communication system is a LAN or WAN?
11. What is an internet? What is the Internet?
12. Why are protocols needed?
13. Why are standards needed?

Exercises

14. What is the maximum number of characters or symbols that can be represented by Unicode?
15. A color image uses 16 bits to represent a pixel. What is the maximum number of different colors that can be represented?
16. Assume six devices are arranged in a mesh topology. How many cables are needed? How many ports are needed for each device?
17. For each of the following four networks, discuss the consequences if a connection fails.
    a. Five devices arranged in a mesh topology
    b. Five devices arranged in a star topology (not counting the hub)
    c. Five devices arranged in a bus topology
    d. Five devices arranged in a ring topology

18. You have two computers connected by an Ethernet hub at home. Is this a LAN, a MAN, or a WAN? Explain your reason.
19. In the ring topology in Figure 1.8, what happens if one of the stations is unplugged?
20. **In** the bus topology in Figure 1.7, what happens if one ofthe stations is unplugged?
21. Draw a hybrid topology with a star backbone and three ring networks.
22. Draw a hybrid topology with a ring backbone and two bus networks.
23. Performance is inversely related to delay. When you use the Internet, which of the following applications are more sensitive to delay?
    a. Sending an e-mail
    b. Copying a file
    c. Surfing the Internet
24. When a party makes a local telephone call to another party, is this a point-to-point or multipoint connection? Explain your answer.
25. Compare the telephone network and the Internet. What are the similarities? What are the differences?

Research Activities

26. Using the site \\iww.cne.gmu.edu/modules/network/osi.html, discuss the OSI model.
27. Using the site www.ansi.org, discuss ANSI's activities.
28. Using the site www.ieee.org, discuss IEEE's activities.
29. Using the site www.ietf.org/, discuss the different types of RFCs.

CHAPTER 2

Network Models

A network is a combination of hardware and software that sends data from one location to another. The hardware consists of the physical equipment that carries signals from one point of the network to another. The software consists of instruction sets that make possible the services that we expect from a network.

We can compare the task of networking to the task of solving a mathematics problem with a computer. The fundamental job of solving the problem with a computer is done by computer hardware. However, this is a very tedious task if only hardware is involved. We would need switches for every memory location to store and manipulate data. The task is much easier if software is available. At the highest level, a program can direct the problem-solving process; the details of how this is done by the actual hardware can be left to the layers of software that are called by the higher levels.

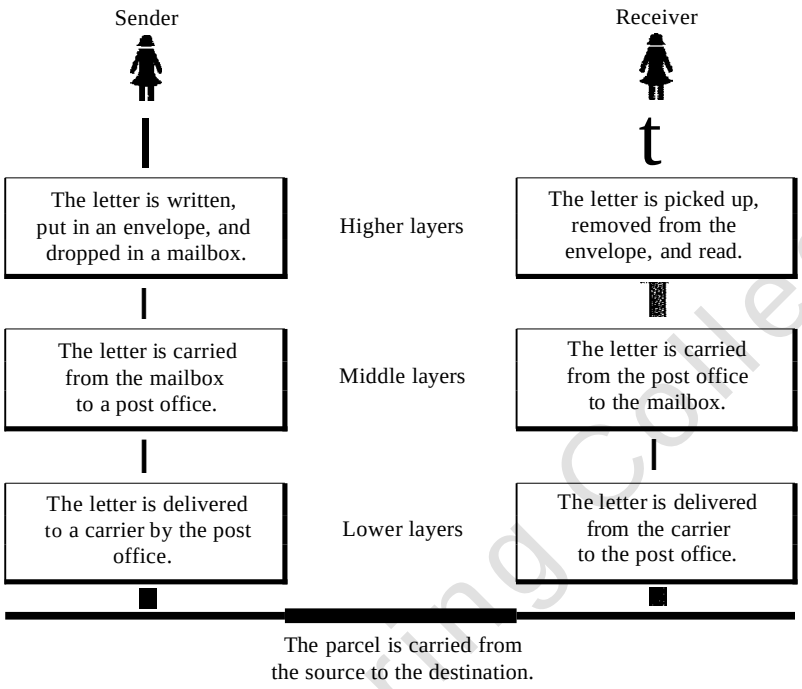
Compare this to a service provided by a computer network. For example, the task of sending an e-mail from one point in the world to another can be broken into several tasks, each performed by a separate software package. Each software package uses the services of another software package. At the lowest layer, a signal, or a set of signals, is sent from the source computer to the destination computer.

In this chapter, we give a general idea of the layers of a network and discuss the functions of each. Detailed descriptions of these layers follow in later chapters.

2.1 LAYERED TASKS

We use the concept of layers in our daily life. As an example, let us consider two friends who communicate through postal maiL The process of sending a letter to a friend would be complex if there were no services available from the post office. Figure 2.1 shows the steps in this task.

Figure 2.1 *Tasks involved in sending a letter*



Sender, Receiver, and Carrier

In Figure 2.1 we have a sender, a receiver, and a carrier that transports the letter. There is a hierarchy of tasks.

At the Sellder Site

Let us first describe, in order, the activities that take place at the sender site.

- Higher layer. The sender writes the letter, inserts the letter in an envelope, writes the sender and receiver addresses, and drops the letter in a mailbox.
- Middle layer. The letter is picked up by a letter carrier and delivered to the post office.
- Lower layer. The letter is sorted at the post office; a carrier transports the letter.

011 the Way

The letter is then on its way to the recipient. On the way to the recipient's local post office, the letter may actually go through a central office. In addition, it may be transported by truck, train, airplane, boat, or a combination of these.

At the Receiver Site

- Lower layer. The carrier transports the letter to the post office.
- Middle layer. The letter is sorted and delivered to the recipient's mailbox.
- Higher layer. The receiver picks up the letter, opens the envelope, and reads it.

Hierarchy

According to our analysis, there are three different activities at the sender site and another three activities at the receiver site. The task of transporting the letter between the sender and the receiver is done by the carrier. Something that is not obvious immediately is that the tasks must be done in the order given in the hierarchy. At the sender site, the letter must be written and dropped in the mailbox before being picked up by the letter carrier and delivered to the post office. At the receiver site, the letter must be dropped in the recipient mailbox before being picked up and read by the recipient.

Services

Each layer at the sending site uses the services of the layer immediately below it. The sender at the higher layer uses the services of the middle layer. The middle layer uses the services of the lower layer. The lower layer uses the services of the carrier.

The layered model that dominated data communications and networking literature before 1990 was the Open Systems Interconnection (OSI) model. Everyone believed that the OSI model would become the ultimate standard for data communications, but this did not happen. The TCPIIP protocol suite became the dominant commercial architecture because it was used and tested extensively in the Internet; the OSI model was never fully implemented.

In this chapter, first we briefly discuss the OSI model, and then we concentrate on TCPIIP as a protocol suite.

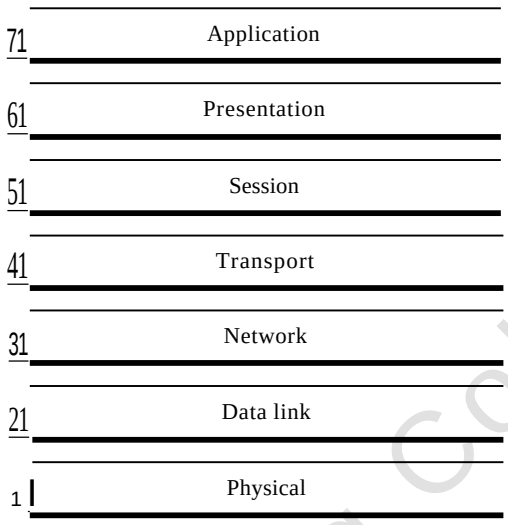
2.2 THE OSI MODEL

Established in 1947, the International Standards Organization (ISO) is a multinational body dedicated to worldwide agreement on international standards. An ISO standard that covers all aspects of network communications is the Open Systems Interconnection model. It was first introduced in the late 1970s. An open system is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture. The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software. The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable.

ISO is the organization. OSI is the model.

The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems. It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network (see Figure 2.2). An understanding of the fundamentals of the OSI model provides a solid basis for exploring data communications.

Figure 2.2 *Seven layers of the OSI model*



Layered **Architecture**

The OSI model is composed of seven ordered layers: physical (layer 1), data link (layer 2), network (layer 3), transport (layer 4), session (layer 5), presentation (layer 6), and application (layer 7). Figure 2.3 shows the layers involved when a message is sent from device A to device B. As the message travels from A to B, it may pass through many intermediate nodes. These intermediate nodes usually involve only the first three layers of the OSI model.

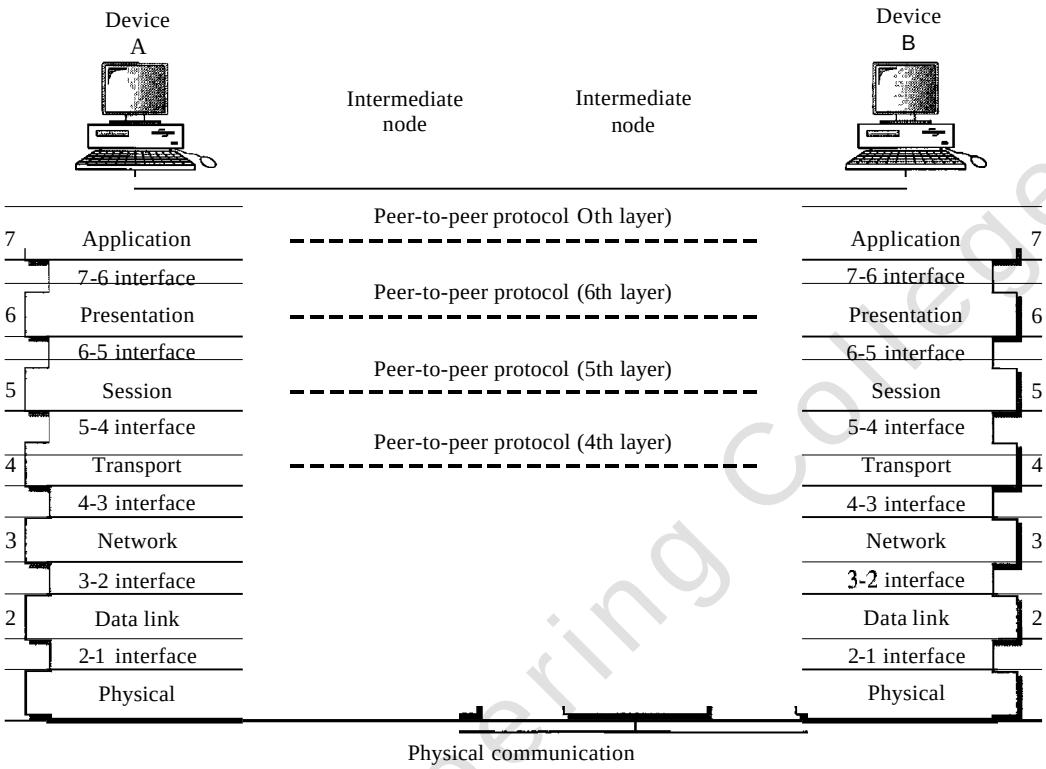
In *developing* the model, the designers distilled the process of transmitting data to its most fundamental elements. They identified which networking functions had related uses and collected those functions into discrete groups that became the layers. Each layer defines a family of functions distinct from those of the other layers. By defining and localizing functionality in this fashion, the designers created an architecture that is both comprehensive and flexible. Most importantly, the OSI model allows complete interoperability between otherwise incompatible systems.

Within a single machine, each layer calls upon the services of the layer just below it. Layer 3, for example, uses the services provided by layer 2 and provides services for layer 4. Between machines, layer *x* on one machine communicates with layer *x* on another machine. This communication is governed by an agreed-upon series of rules and conventions called protocols. The processes on each machine that communicate at a given layer are called peer-to-peer processes. Communication between machines is therefore a peer-to-peer process using the protocols appropriate to a given layer.

Peer-to-Peer Processes

At the physical layer, communication is direct: In Figure 2.3, device A sends a stream of bits to device B (through intermediate nodes). At the higher layers, however, communication must move down through the layers on device A, over to device B, and then

Figure 2.3 *The interaction between layers in the OSI model*



back up through the layers. Each layer in the sending device adds its own information to the message it receives from the layer just above it and passes the whole package to the layer just below it.

At layer I the entire package is converted to a form that can be transmitted to the receiving device. At the receiving machine, the message is unwrapped layer by layer, with each process receiving and removing the data meant for it. For example, layer 2 removes the data meant for it, then passes the rest to layer 3. Layer 3 then removes the data meant for it and passes the rest to layer 4, and so on.

Interfaces Between Layers

The passing of the data and network information down through the layers of the sending device and back up through the layers of the receiving device is made possible by an interface between each pair of adjacent layers. Each interface defines the information and services a layer must provide for the layer above it. Well-defined interfaces and layer functions provide modularity to a network. As long as a layer provides the expected services to the layer above it, the specific implementation of its functions can be modified or replaced without requiring changes to the surrounding layers.

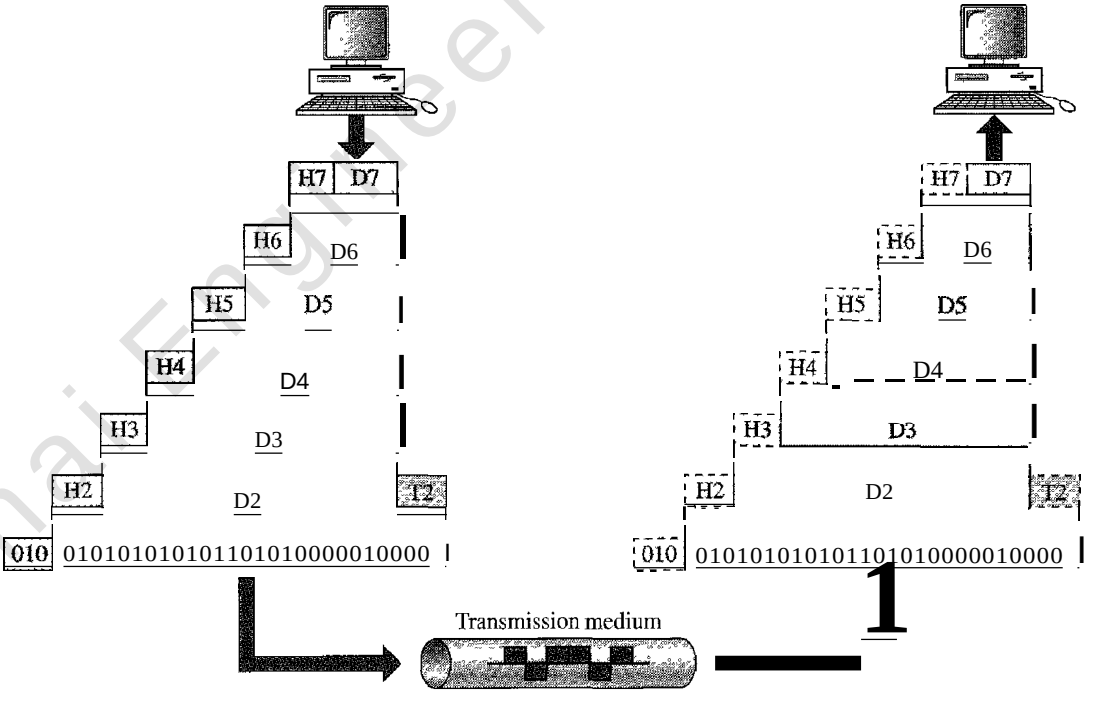
Organization of the Layers

The seven layers can be thought of as belonging to three subgroups. Layers I, 2, and 3-physical, data link, and network-are the network support layers; they deal with

the physical aspects of moving data from one device to another (such as electrical specifications, physical connections, physical addressing, and transport timing and reliability). Layers 5, 6, and 7-session, presentation, and application-can be thought of as the user support layers; they allow interoperability among unrelated software systems. Layer 4, the transport layer, links the two subgroups and ensures that what the lower layers have transmitted is in a form that the upper layers can use. The upper OSI layers are almost always implemented in software; lower layers are a combination of hardware and software, except for the physical layer, which is mostly hardware.

In Figure 2.4, which gives an overall view of the OSI layers, D7 means the data unit at layer 7, D6 means the data unit at layer 6, and so on. The process starts at layer 7 (the application layer), then moves from layer to layer in descending, sequential order. At each layer, a **header,** or possibly a **trailer,** can be added to the data unit. Commonly, the trailer is added only at layer 2. When the formatted data unit passes through the physical layer (layer 1), it is changed into an electromagnetic signal and transported along a physical link.

Figure 2.4 *An exchange using the OS! model*



Upon reaching its destination, the signal passes into layer 1 and is transformed back into digital form. The data units then move back up through the OSI layers. As each block of data reaches the next higher layer, the headers and trailers attached to it at the corresponding sending layer are removed, and actions appropriate to that layer are taken. By the time it reaches layer 7, the message is again in a form appropriate to the application and is made available to the recipient.

Encapsulation

Figure 2.3 reveals another aspect of data communications in the OSI model: encapsulation. A packet (header and data) at level 7 is encapsulated in a packet at level 6. The whole packet at level 6 is encapsulated in a packet at level 5, and so on.

In other words, the data portion of a packet at level $N - 1$ carries the whole packet (data and header and maybe trailer) from level N. The concept is called *encapsulation;* level $N - 1$ is not aware of which part of the encapsulated packet is data and which part is the header or trailer. For level $N - 1$, the whole packet coming from level N is treated as one integral unit.

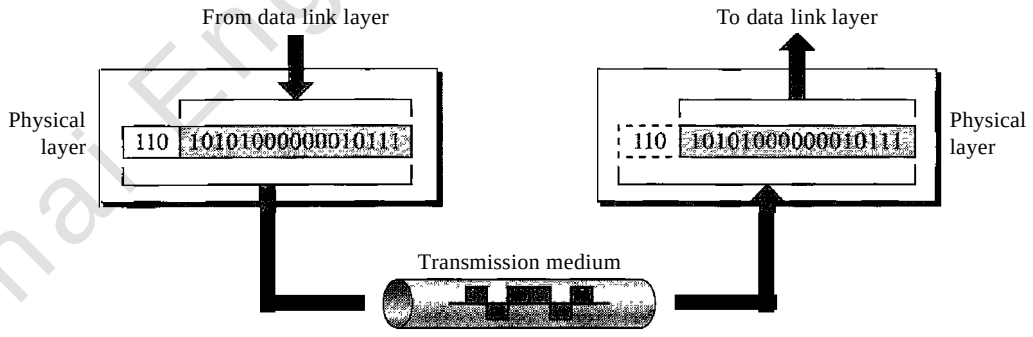
2.3 LAYERS IN THE OSI MODEL

In this section we briefly describe the functions of each layer in the OSI model.

Physical Layer

The physical layer coordinates the functions required to carry a bit stream over a physical medium. It deals with the mechanical and electrical specifications of the interface and transmission medium. It also defines the procedures and functions that physical devices and interfaces have to perform for transmission to Occur. Figure 2.5 shows the position of the physical layer with respect to the transmission medium and the data link layer.

Figure 2.5 *Physical layer*



The physical layer is responsible for movements of individual bits from one hop (node) to the next.

The physical layer is also concerned with the following:

- Physical characteristics of interfaces and medium. The physical layer defines the characteristics of the interface between the devices and the transmission medium. It also defines the type of transmission medium.
- Representation of bits. The physical layer data consists of a stream of bits (sequence of Os or 1s) with no interpretation. To be transmitted, bits must be

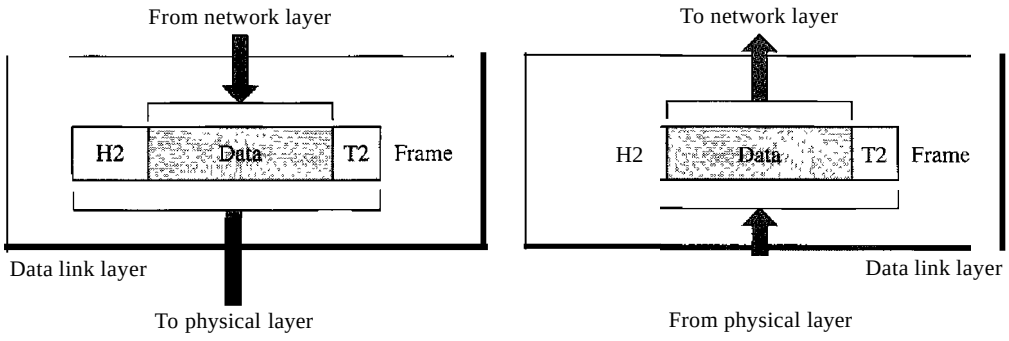
encoded into signals--electrical or optical. The physical layer defines the type of encoding (how 0s and 1s are changed to signals).

- Data rate. The transmission rate-the number of bits sent each second-is also defined by the physical layer. In other words, the physical layer defines the duration of a bit, which is how long it lasts.
- Synchronization of bits. The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.
- Line configuration. The physical layer is concerned with the connection of devices to the media. In a point-to-point configuration, two devices are connected through a dedicated link. In a multipoint configuration, a link is shared among several devices.
- Physical topology. The physical topology defines how devices are connected to make a network. Devices can be connected by using a mesh topology (every device is connected to every other device), a star topology (devices are connected through a central device), a ring topology (each device is connected to the next, forming a ring), a bus topology (every device is on a common link), or a hybrid topology (this is a combination of two or more topologies).
- Transmission mode. The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex. In simplex mode, only one device can send; the other can only receive. The simplex mode is a one-way communication. In the half-duplex mode, two devices can send and receive, but not at the same time. In a full-duplex (or simply duplex) mode, two devices can send and receive at the same time.

Data Link Layer

The data link layer transforms the physical layer, a raw transmission facility, to a reliable link. It makes the physical layer appear error-free to the upper layer (network layer). Figure 2.6 shows the relationship of the data link layer to the network and physicallayers.

Figure 2.6 *Data link layer*



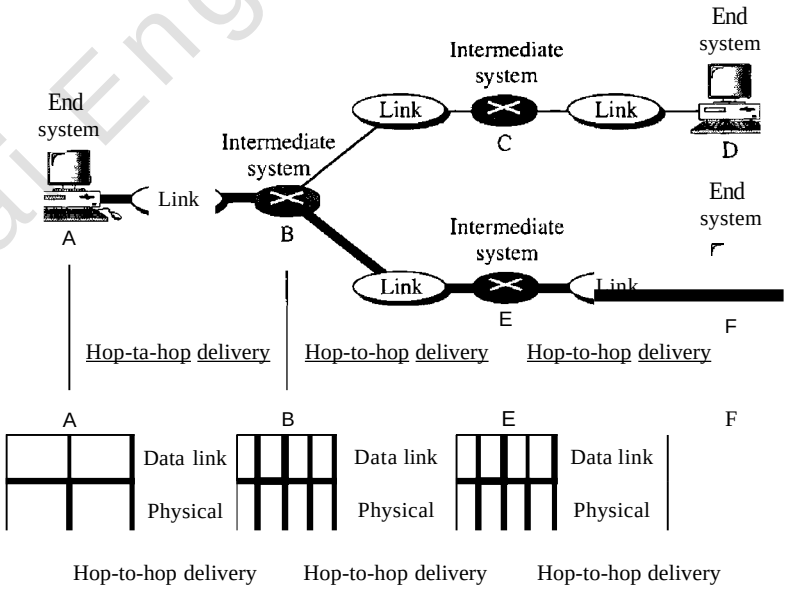
The data link layer is responsible for moving frames from one hop (node) to the next.

Other responsibilities of the data link layer include the following:

- Framing. The data link layer divides the stream of bits received from the network layer into manageable data units called frames.
- Physical addressing. If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects the network to the next one.
- Flow control. If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
- Error control. The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplicate frames. Error control is normally achieved through a trailer added to the end of the frame.
- Access control. When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

Figure 2.7 illustrates hop-to-hop (node-to-node) delivery by the data link layer.

Figure 2.7 *Hop-to-hop delivery*



As the figure shows, communication at the data link layer occurs between two adjacent nodes. To send data from A to F, three partial deliveries are made. First, the data link layer at A sends a frame to the data link layer at B (a router). Second, the data

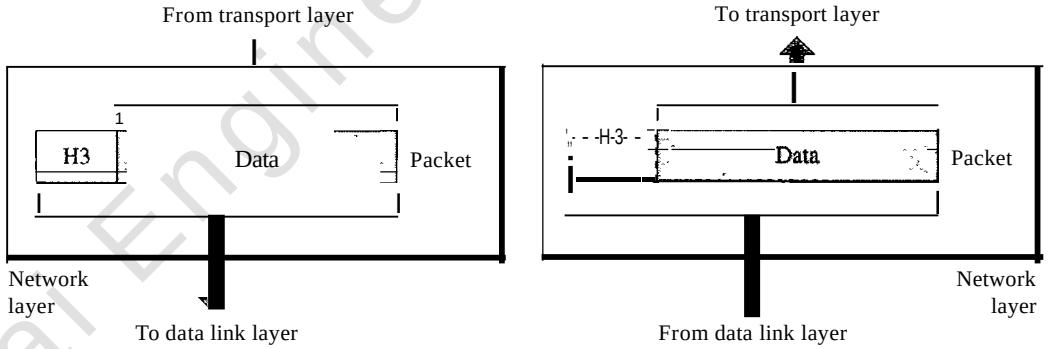
link layer at B sends a new frame to the data link layer at E. Finally, the data link layer at E sends a new frame to the data link layer at F. Note that the frames that are exchanged between the three nodes have different values in the headers. The frame from A to B has B as the destination address and A as the source address. The frame from B to E has E as the destination address and B as the source address. The frame from E to F has F as the destination address and E as the source address. The values of the trailers can also be different if error checking includes the header of the frame.

Network Layer

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination.

If two systems are connected to the same link, there is usually no need for a network layer. However, if the two systems are attached to different networks (links) with connecting devices between the networks (links), there is often a need for the network layer to accomplish source-to-destination delivery. Figure 2.8 shows the relationship of the network layer to the data link and transport layers.

Figure 2.8 *Network layer*



The network layer is responsible for the delivery of individual packets from the source host to the destination host.

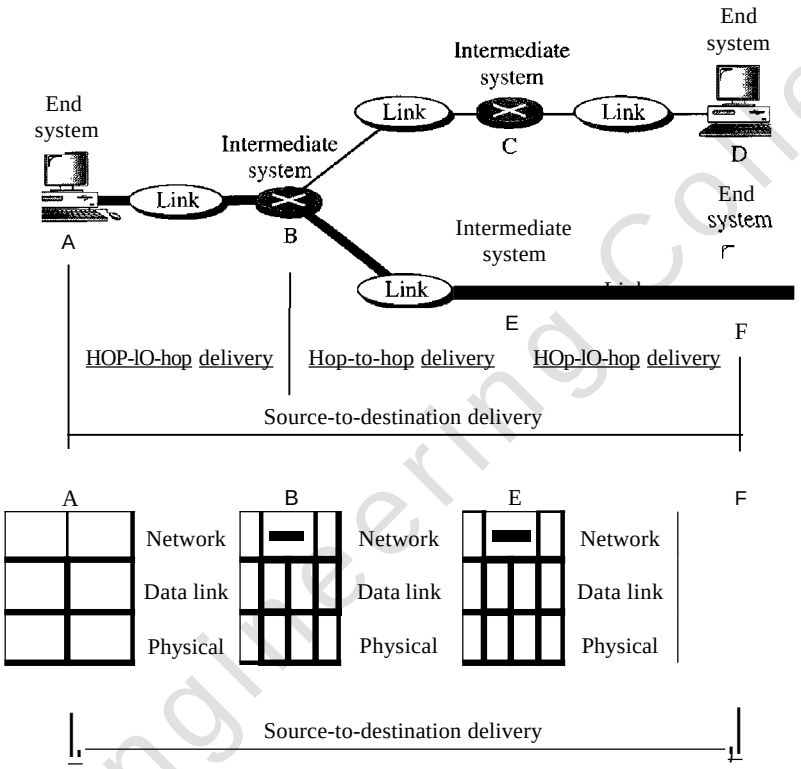
Other responsibilities of the network layer include the following:

- Logical addressing. The physical addressing implemented by the data link layer handles the addressing problem locally. If a packet passes the network boundary, we need another addressing system to help distinguish the source and destination systems. The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver. We discuss logical addresses later in this chapter.
- Routing. When independent networks or links are connected to create *intemetworks* (network of networks) or a large network, the connecting devices (called *routers*

or *switches)* route or switch the packets to their final destination. One of the functions of the network layer is to provide this mechanism.

Figure 2.9 illustrates end-to-end delivery by the network layer.

Figure 2.9 *Source-to-destination delivery*

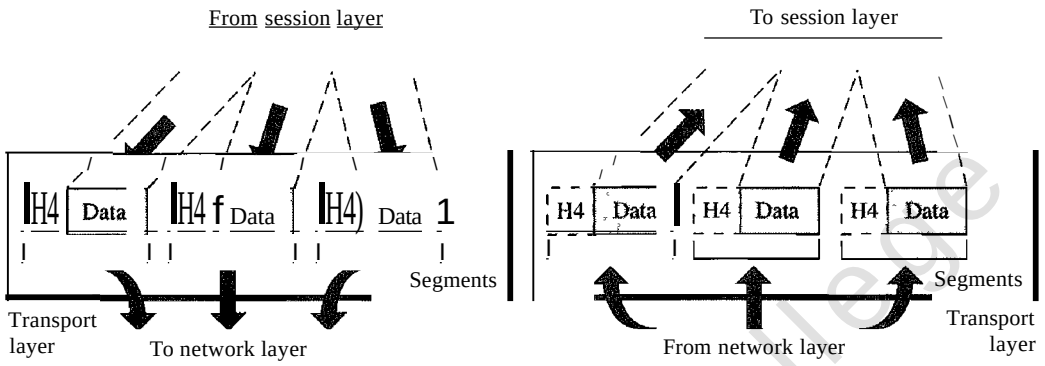


As the figure shows, now we need a source-to-destination delivery. The network layer at A sends the packet to the network layer at B. When the packet arrives at router B, the router makes a decision based on the final destination (F) of the packet. As we will see in later chapters, router B uses its routing table to find that the next hop is router E. The network layer at B, therefore, sends the packet to the network layer at E. The network layer at E, in tum, sends the packet to the network layer at F.

Transport Layer

The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on a host. Whereas the network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level. Figure 2.10 shows the relationship of the transport layer to the network and session layers.

Figure 2.10 *Transport layer*



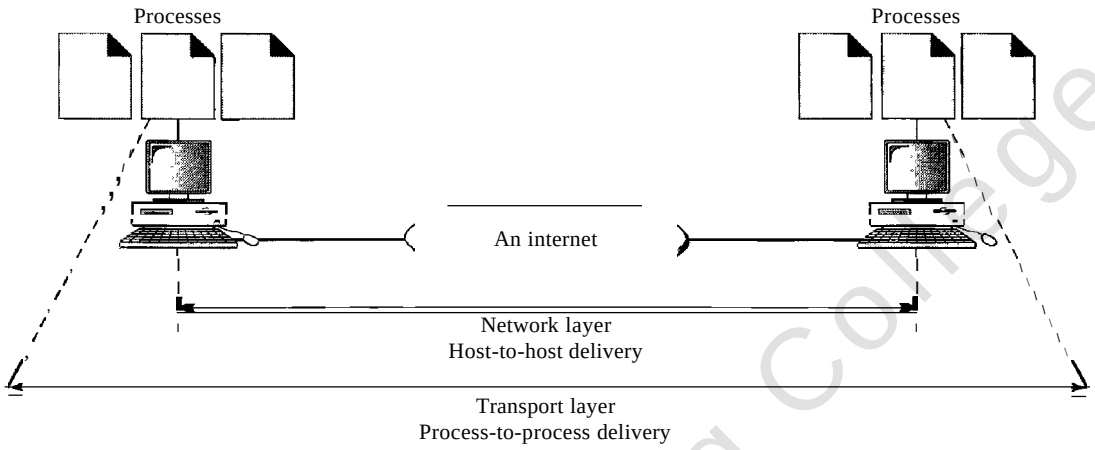
The transport layer is responsible for the delivery of a message from one process to another.

Other responsibilities of the transport layer include the following:

- Service-point addressing. Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other. The transport layer header must therefore include a type of address called a *service-point address* (or port address). The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on that computer.
- Segmentation and reassembly. A message is divided into transmittable segments, with each segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.
- Connection control. The transport layer can be either connectionless or connection-oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.
- Flow control. Like the data link layer, the transport layer is responsible for flow control. However, flow control at this layer is performed end to end rather than across a single link.
- Error control. Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed process-to-process rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication). Error correction is usually achieved through retransmission.

Figure 2.11 illustrates process-to-process delivery by the transport layer.

Figure 2.11 *Reliable process-to-process delivery of a message*



Session Layer

The services provided by the first three layers (physical, data link, and network) are not sufficient for some processes. The session layer is the network *dialog controller.* It establishes, maintains, and synchronizes the interaction among communicating systems.

The session layer is responsible for dialog control and synchronization.

Specific responsibilities of the session layer include the following:

- Dialog control. The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half-duplex (one way at a time) or full-duplex (two ways at a time) mode.
- Synchronization. The session layer allows a process to add checkpoints, or synChronization points, to a stream of data. For example, if a system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be resent. Figure 2.12 illustrates the relationship of the session layer to the transport and presentation layers.

Presentation Layer

The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems. Figure 2.13 shows the relationship between the presentation layer and the application and session layers.

Figure 2.12 *Session layer*

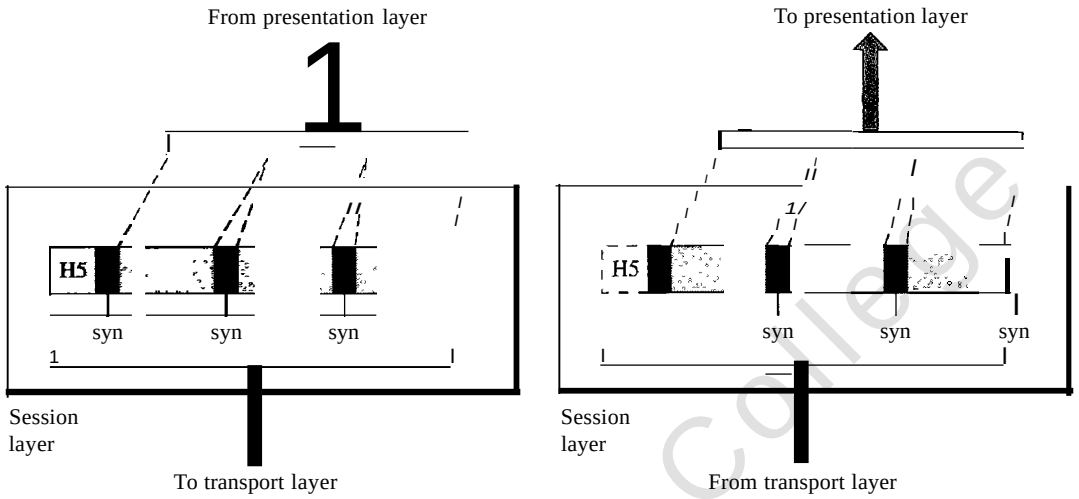
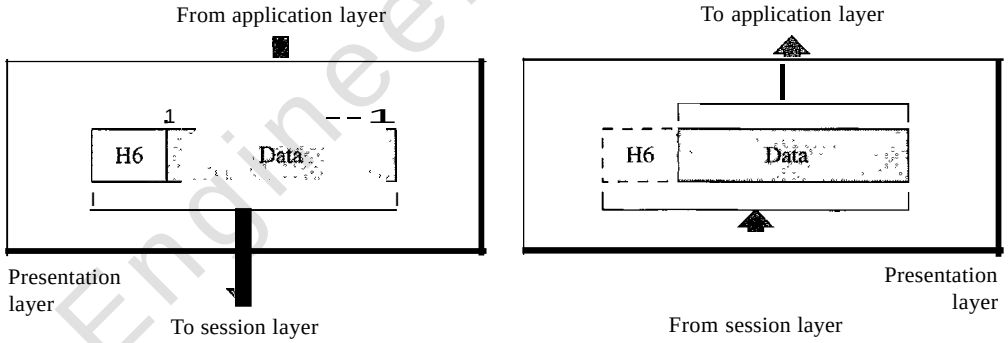


Figure 2.13 *Presentation layer*



The presentation layer is responsible for translation, compression, and encryption.

Specific responsibilities of the presentation layer include the following:

- Translation. The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. The infonnation must be changed to bit streams before being transmitted. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the information from its sender-dependent format into a common format. The presentation layer at the receiving machine changes the common format into its receiver-dependent format.
- Encryption. To carry sensitive information, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to

another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.

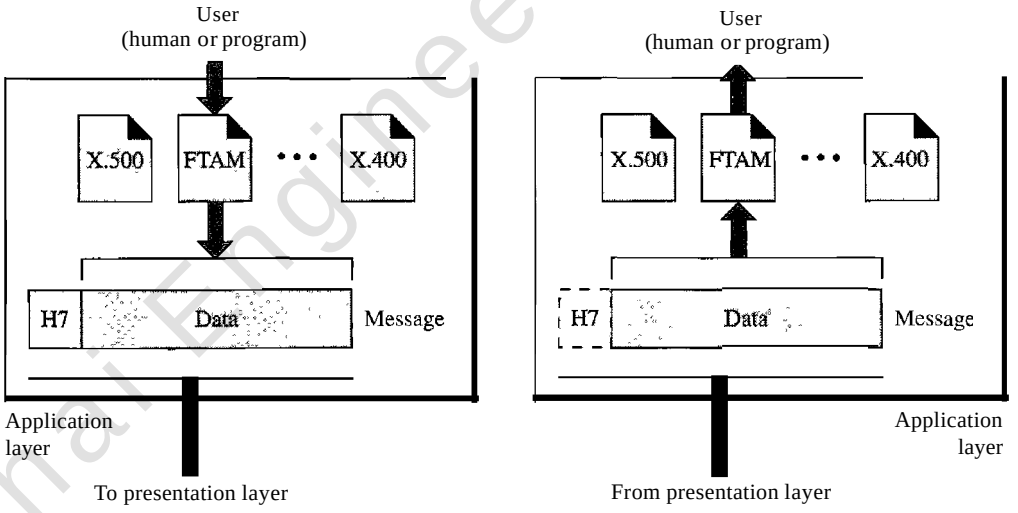
- Compression. Data compression reduces the number of bits contained in the information. Data compression becomes particularly important in the transmission of multimedia such as text, audio, and video.

Application Layer

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services.

Figure 2.14 shows the relationship of the application layer to the user and the presentation layer. Of the many application services available, the figure shows only three: *XAOO* (message-handling services), X.500 (directory services), and file transfer, access, and management (FTAM). The user in this example employs *XAOO* to send an e-mail message.

Figure 2.14 *Application layer*



The application layer is responsible for providing services to the user.

Specific services provided by the application layer include the following:

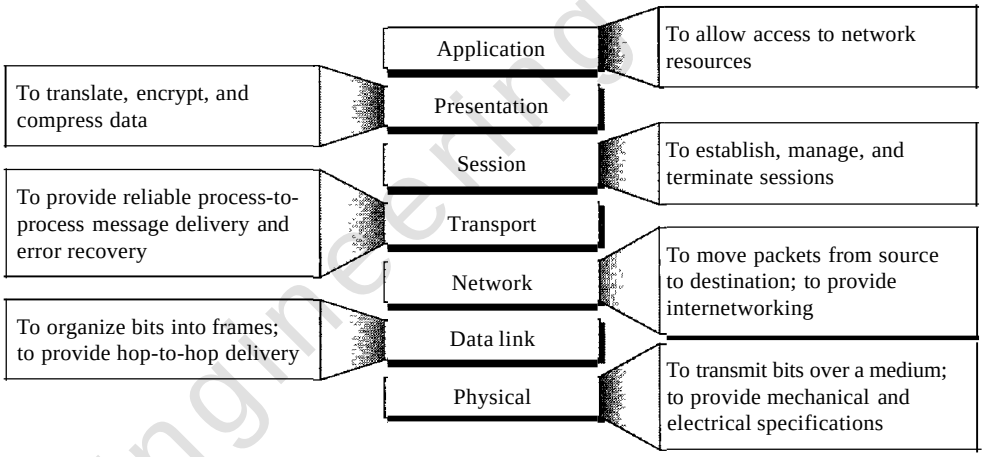
- Network virtual terminal. A network virtual terminal is a software version of a physical terminal, and it allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user's computer talks to the software terminal which, in turn, talks to the host, and vice versa. The remote host believes it is communicating with one of its own terminals and allows the user to log on.

- File transfer, access, and management. This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.
- Mail services. This application provides the basis for e-mail forwarding and storage.
- Directory services. This application provides distributed database sources and access for global information about various objects and services.

Summary of Layers

Figure 2.15 shows a summary of duties for each layer.

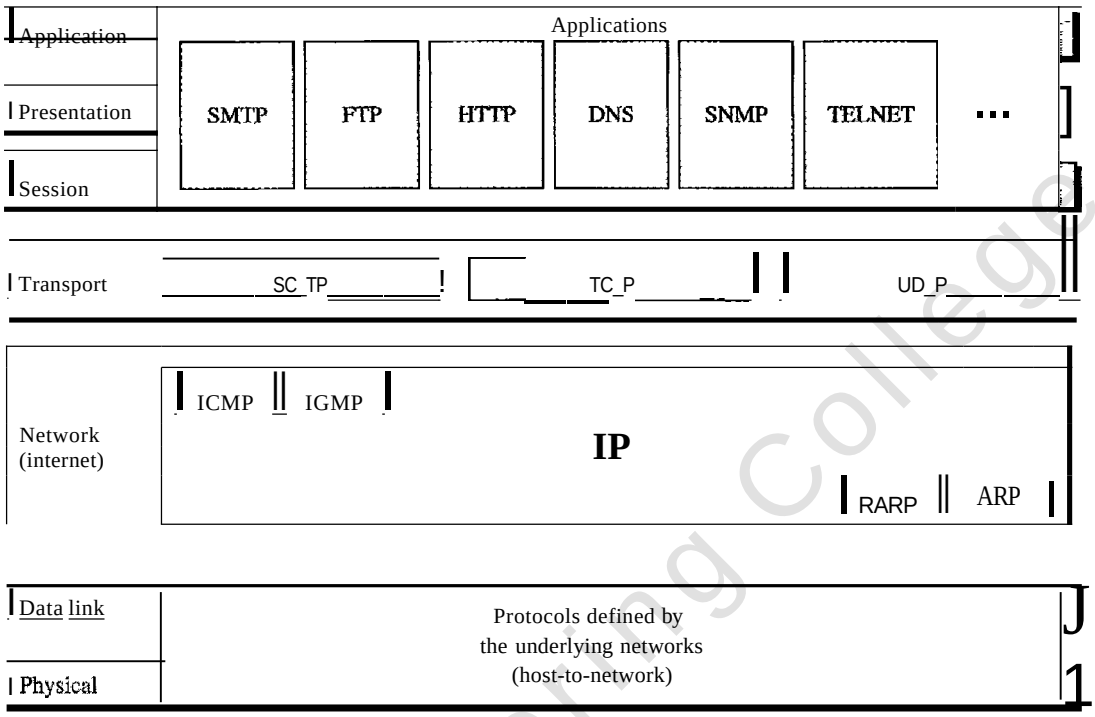
Figure 2.15 *Summary of layers*



2.4 TCP/IP PROTOCOL SUITE

The TCPIIP protocol suite was developed prior to the OSI model. Therefore, the layers in the TCP/IP protocol suite do not exactly match those in the OSI model. The original TCP/IP protocol suite was defined as having four layers: host-to-network, internet, transport, and application. However, when TCP/IP is compared to OSI, we can say that the host-to-network layer is equivalent to the combination of the physical and data link layers. The internet layer is equivalent to the network layer, and the application layer is roughly doing the job of the session, presentation, and application layers with the transport layer in TCPIIP taking care of part of the duties of the session layer. So in this book, we assume that the TCPIIP protocol suite is made of five layers: physical, data link, network, transport, and application. The first four layers provide physical standards, network interfaces, internetworking, and transport functions that correspond to the first four layers of the OSI model. The three topmost layers in the OSI model, however, are represented in TCPIIP by a single layer called the *application layer* (see Figure 2.16).

Figure 2.16 *TCPIIP and OSI model*



TCP/IP is a hierarchical protocol made up of interactive modules, each of which provides a specific functionality; however, the modules are not necessarily interdependent. Whereas the OSI model specifies which functions belong to each of its layers, the layers of the *TCP/IP* protocol suite contain relatively independent protocols that can be mixed and matched depending on the needs of the system. The term *hierarchical* means that each upper-level protocol is supported by one or more lower-level protocols.

At the transport layer, *TCP/IP* defines three protocols: Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP). At the network layer, the main protocol defined by TCP/IP is the Internetworking Protocol (IP); there are also some other protocols that support data movement in this layer.

Physical and Data Link Layers

At the physical and data link layers, *TCPIIP* does not define any specific protocol. It supports all the standard and proprietary protocols. A network in a *TCPIIP* internetwork can be a local-area network or a wide-area network.

Network Layer

At the network layer (or, more accurately, the internetwork layer), *TCP/IP* supports the Internetworking Protocol. IP, in turn, uses four supporting protocols: ARP, RARP, ICMP, and IGMP. Each of these protocols is described in greater detail in later chapters.

Internetworking Protocol (IP)

The Internetworking Protocol (IP) is the transmission mechanism used by the TCP/IP protocols. It is an unreliable and connectionless protocol-a best-effort delivery service. The term *best effort* means that IP provides no error checking or tracking. IP assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.

IP transports data in packets called *datagrams,* each of which is transported separately. Datagrams can travel along different routes and can arrive out of sequence or be duplicated. IP does not keep track of the routes and has no facility for reordering datagrams once they arrive at their destination.

The limited functionality of IP should not be considered a weakness, however. IP provides bare-bones transmission functions that free the user to add only those facilities necessary for a given application and thereby allows for maximum efficiency. IP is discussed in Chapter 20.

Address Resolution Protocol

The Address Resolution Protocol (ARP) is used to associate a logical address with a physical address. On a typical physical network, such as a LAN, each device on a link is identified by a physical or station address, usually imprinted on the network interface card (NIC). ARP is used to find the physical address of the node when its Internet address is known. ARP is discussed in Chapter 21.

Reverse Address Resolution Protocol

The Reverse Address Resolution Protocol (RARP) allows a host to discover its Internet address when it knows only its physical address. It is used when a computer is connected to a network for the first time or when a diskless computer is booted. We discuss RARP in Chapter 21.

Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) is a mechanism used by hosts and gateways to send notification of datagram problems back to the sender. ICMP sends query and error reporting messages. We discuss ICMP in Chapter 21.

Internet Group Message Protocol

The Internet Group Message Protocol (IGMP) is used to facilitate the simultaneous transmission of a message to a group of recipients. We discuss IGMP in Chapter 22.

Transport Layer

Traditionally the transport layer was represented in *TCP/IP* by two protocols: TCP and UDP. IP is a host-to-host protocol, meaning that it can deliver a packet from one physical device to another. UDP and TCP are transport level protocols responsible for delivery of a message from a process (running program) to another process. A new transport layer protocol, SCTP, has been devised to meet the needs of some newer applications.

User Datagram Protocol

The User Datagram Protocol (UDP) is the simpler of the two standard TCPIIP transport protocols. It is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from the upper layer. UDP is discussed in Chapter 23.

Transmission Control Protocol

The Transmission Control Protocol (TCP) provides full transport-layer services to applications. TCP is a reliable stream transport protocol. The term *stream,* in this context, means connection-oriented: A connection must be established between both ends of a transmission before either can transmit data.

At the sending end of each transmission, TCP divides a stream of data into smaller units called *segments.* Each segment includes a sequence number for reordering after receipt, together with an acknowledgment number for the segments received. Segments are carried across the internet inside of IP datagrams. At the receiving end, TCP collects each datagram as it comes in and reorders the transmission based on sequence numbers. TCP is discussed in Chapter 23.

Stream Control Transmission Protocol

The Stream Control Transmission Protocol (SCTP) provides support for newer applications such as voice over the Internet. It is a transport layer protocol that combines the best features of UDP and TCP. We discuss SCTP in Chapter 23.

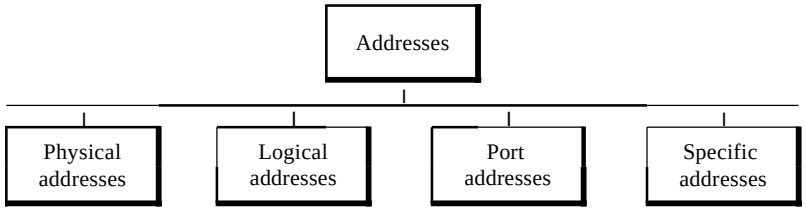
Application Layer

The *application layer* in TCPIIP is equivalent to the combined session, presentation, and application layers in the OSI modeL Many protocols are defined at this layer. We cover many of the standard protocols in later chapters.

2.5 ADDRESSING

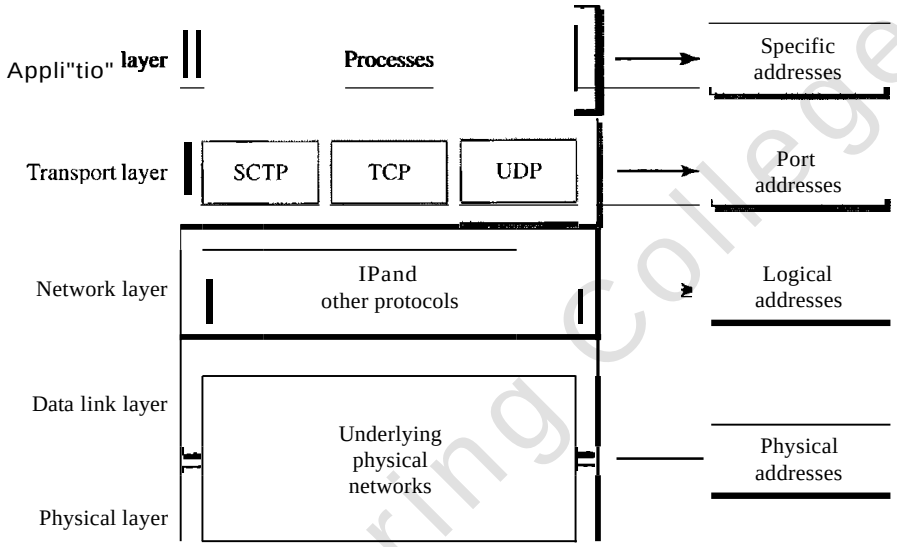
Four levels of addresses are used in an internet employing the *TCP/IP* protocols: physical (link) addresses, logical (IP) addresses, port addresses, and specific addresses (see Figure 2.17).

Figure 2.17 *Addresses in TCPIIP*



Each address is related to a specific layer in the TCPIIP architecture, as shown in Figure 2.18.

Figure 2.18 *Relationship of layers and addresses in TCPIIP*



Physical Addresses

The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN. It is included in the frame used by the data link layer. It is the lowest-level address.

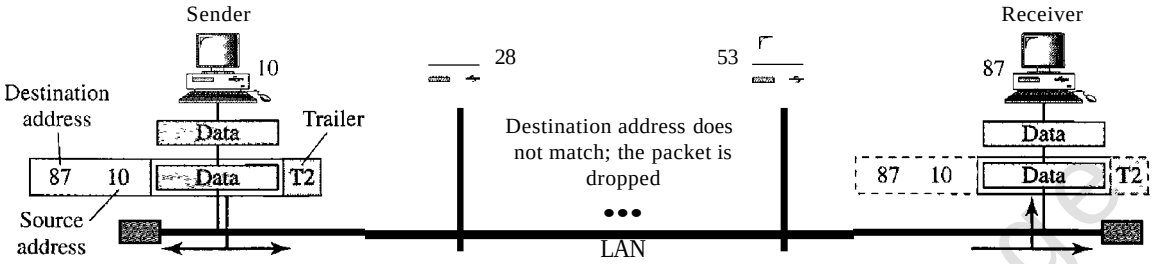
The physical addresses have authority over the network (LAN or WAN). The size and format of these addresses vary depending on the network. For example, Ethernet uses a 6-byte (48-bit) physical address that is imprinted on the network interface card (NIC). LocalTalk (Apple), however, has a I-byte dynamic address that changes each time the station comes up.

Example 2.1

In Figure 2.19 a node with physical address 10 sends a frame to a node with physical address 87. The two nodes are connected by a link (bus topology LAN). At the data link layer, this frame contains physical (link) addresses in the header. These are the only addresses needed. The rest of the header contains other information needed at this level. The trailer usually contains extra bits needed for error detection. As the figure shows, the computer with physical address lOis the sender, and the computer with physical address 87 is the receiver. The data link layer at the sender receives data from an upper layer. It encapsulates the data in a frame, adding a header and a trailer. The header, among other pieces of information, carries the receiver and the sender physical (link) addresses. Note that in most data link protocols, the destination address, 87 in this case, comes before the source address (10 in this case).

We have shown a bus topology for an isolated LAN. In a bus topology, the frame is propagated in both directions (left and right). The frame propagated to the left dies when it reaches the end of the cable if the cable end is terminated appropriately. The frame propagated to the right is

Figure 2.19 *Physical addresses*



sent to every station on the network. Each station with a physical addresses other than 87 drops the frame because the destination address in the frame does not match its own physical address. The intended destination computer, however, finds a match between the destination address in the frame and its own physical address. The frame is checked, the header and trailer are dropped, and the data part is decapsulated and delivered to the upper layer.

Example 2.2

As we will see in Chapter 13, most local-area networks use a 48-bit (6-byte) physical address written as 12 hexadecimal digits; every byte (2 hexadecimal digits) is separated by a colon, as shown below:

07:01:02:01:2C:4B
A 6-byte (12 hexadecimal digits) physical address

Logical Addresses

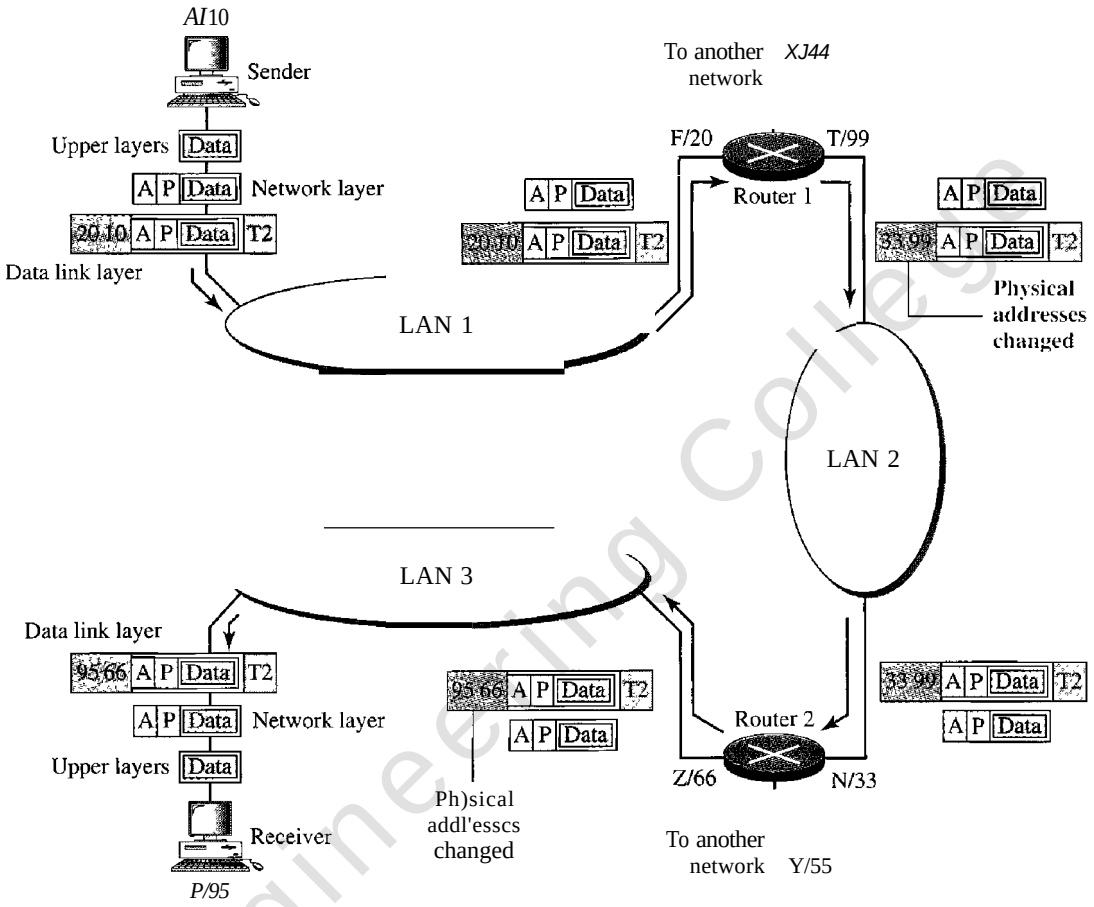
Logical addresses are necessary for universal communications that are independent of underlying physical networks. Physical addresses are not adequate in an internetwork environment where different networks can have different address formats. A universal addressing system is needed in which each host can be identified uniquely, regardless of the underlying physical network.

The logical addresses are designed for this purpose. A logical address in the Internet is currently a 32-bit address that can uniquely define a host connected to the Internet. No two publicly addressed and visible hosts on the Internet can have the same IP address.

Example 2.3

Figure 2.20 shows a part of an internet with two routers connecting three LANs. Each device (computer or router) has a pair of addresses (logical and physical) for each connection. In this case, each computer is connected to only one link and therefore has only one pair of addresses. Each router, however, is connected to three networks (only two are shown in the figure). So each router has three pairs of addresses, one for each connection. Although it may obvious that each router must have a separate physical address for each connection, it may not be obvious why it needs a logical address for each connection. We discuss these issues in Chapter 22 when we discuss routing.

Figure 2.20 *IP addresses*



The computer with logical address A and physical address 10 needs to send a packet to the computer with logical address P and physical address 95. We use letters to show the logical addresses and numbers for physical addresses, but note that both are actually numbers, as we will see later in the chapter.

The sender encapsulates its data in a packet at the network layer and adds two logical addresses (A and P). Note that in most protocols, the logical source address comes before the logical destination address (contrary to the order of physical addresses). The network layer, however, needs to find the physical address of the next hop before the packet can be delivered. The network layer consults its routing table (see Chapter 22) and finds the logical address of the next hop (router I) to be F. The ARP discussed previously finds the physical address of router 1 that corresponds to the logical address of 20. Now the network layer passes this address to the data link layer, which in tum, encapsulates the packet with physical destination address 20 and physical source address 10.

The frame is received by every device on LAN 1, but is discarded by all except router 1, which finds that the destination physical address in the frame matches with its own physical address. The router decapsulates the packet from the frame to read the logical destination address P. Since the logical destination address does not match the router's logical address, the router knows that the packet needs to be forwarded. The

router consults its routing table and ARP to find the physical destination address of the next hop (router 2), creates a new frame, encapsulates the packet, and sends it to router 2.

Note the physical addresses in the frame. The source physical address changes from 10 to 99. The destination physical address changes from 20 (router 1 physical address) to 33 (router 2 physical address). The logical source and destination addresses must remain the same; otherwise the packet will be lost.

At router 2 we have a similar scenario. The physical addresses are changed, and a new frame is sent to the destination computer. When the frame reaches the destination, the packet is decapsulated. The destination logical address P matches the logical address of the computer. The data are decapsulated from the packet and delivered to the upper layer. Note that although physical addresses will change from hop to hop, logical addresses remain the same from the source to destination. There are some exceptions to this rule that we discover later in the book.

The physical addresses will change from hop to hop, but the logical addresses usually remain the same.

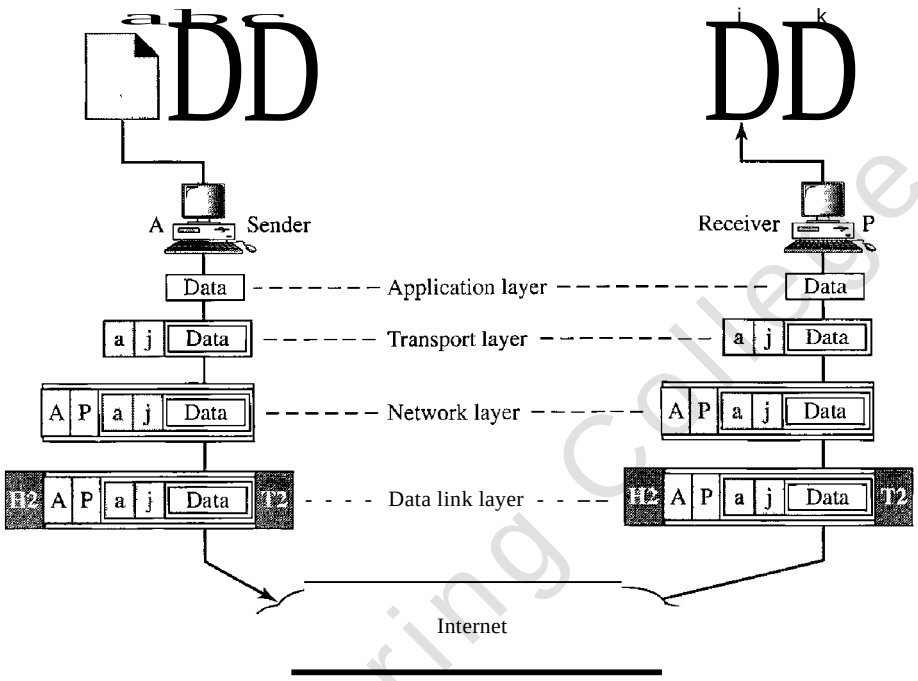
Port Addresses

The IP address and the physical address are necessary for a quantity of data to travel from a source to the destination host. However, arrival at the destination host is not the final objective of data communications on the Internet. A system that sends nothing but data from one computer to another is not complete. Today, computers are devices that can run multiple processes at the same time. The end objective of Internet communication is a process communicating with another process. For example, computer A can communicate with computer C by using TELNET. At the same time, computer A communicates with computer B by using the File Transfer Protocol (FTP). For these processes to receive data simultaneously, we need a method to label the different processes. In other words, they need addresses. In the TCPIIP architecture, the label assigned to a process is called a port address. A port address in TCPIIP is 16 bits in length.

Example 2.4

Figure 2.21 shows two computers communicating via the Internet. The sending computer is running three processes at this time with port addresses a, b, and c. The receiving computer is running two processes at this time with port addresses j and k. Process a in the sending computer needs to communicate with process j in the receiving computer. Note that although both computers are using the same application, FTP, for example, the port addresses are different because one is a client program and the other is a server program, as we will see in Chapter 23. To show that data from process a need to be delivered to process j, and not k, the transport layer encapsulates data from the application layer in a packet and adds two port addresses (a and j), source and destination. The packet from the transport layer is then encapsulated in another packet at the network layer with logical source and destination addresses (A and P). Finally, this packet is encapsulated in a frame with the physical source and destination addresses of the next hop. We have not shown the physical addresses because they change from hop to hop inside the cloud designated as the Internet. Note that although physical addresses change from hop to hop, logical and port addresses remain the same from the source to destination. There are some exceptions to this rule that we discuss later in the book.

Figure 2.21 *Port addresses*



The physical addresses change from hop to hop, but the logical and port addresses usually remain the same.

Example 2.5

As we will see in Chapter 23, a port address is a 16-bit address represented by one decimal number as shown.

753

A 16-bit port address represented as one single number

Specific Addresses

Some applications have user-friendly addresses that are designed for that specific address. Examples include the e-mail address (for example, forouzan@fhda.edu) and the Universal Resource Locator (URL) (for example, www.mhhe.com). The first defines the recipient of an e-mail (see Chapter 26); the second is used to find a document on the World Wide Web (see Chapter 27). These addresses, however, get changed to the corresponding port and logical addresses by the sending computer, as we will see in Chapter 25.

2.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items enclosed in brackets, [...] refer to the reference list at the end of the text.

Books

Network models are discussed in Section 1.3 of [Tan03], Chapter 2 of [For06], Chapter 2 of [Sta04], Sections 2.2 and 2.3 of [GW04], Section 1.3 of [PD03], and Section 1.7 of [KR05]. A good discussion about addresses can be found in Section 1.7 of [Ste94].

Sites

The following site is related to topics discussed in this chapter.

- www.osi.org! Information about OS1.

RFCs

The following site lists all RFCs, including those related to IP and port addresses.

- www.ietLorg/rfc.html

2.7 KEY TERMS

access control
Address Resolution Protocol (ARP)
application layer
best-effort delivery
bits
connection control
data link layer
encoding
error
error control
flow control
frame
header
hop-to-hop delivery
host-to-host protocol
interface
Internet Control Message Protocol (ICMP)
Internet Group Message Protocol (IGMP)
logical addressing
mail service
network layer
node-to-node delivery
open system
Open Systems Interconnection (OSI) model
peer-to-peer process
physical addressing
physical layer
port address
presentation layer
process-to-process delivery
Reverse Address Resolution Protocol (RARP)
routing
segmentation
session layer
source-to-destination delivery
Stream Control Transmission Protocol (SCTP)
synchronization point
TCPIIP protocol suite
trailer
Transmission Control Protocol (TCP)
transmission rate
transport layer
transport level protocols
User Datagram Protocol (UDP)

2.8 SUMMARY

- The International Standards Organization created a model called the Open Systems Interconnection, which allows diverse systems to communicate.
- The seven-layer OSI model provides guidelines for the development of universally compatible networking protocols.
- The physical, data link, and network layers are the network support layers.
- The session, presentation, and application layers are the user support layers.
- The transport layer links the network support layers and the user support layers.
- The physical layer coordinates the functions required to transmit a bit stream over a physical medium.
- The data link layer is responsible for delivering data units from one station to the next without errors.
- The network layer is responsible for the source-to-destination delivery of a packet across multiple network links.
- The transport layer is responsible for the process-to-process delivery of the entire message.
- The session layer establishes, maintains, and synchronizes the interactions between communicating devices.
- The presentation layer ensures interoperability between communicating devices through transformation of data into a mutually agreed upon format.
- The application layer enables the users to access the network.
- TCP/IP is a five-layer hierarchical protocol suite developed before the OSI model.
- The TCP/IP application layer is equivalent to the combined session, presentation, and application layers of the OSI model.
- Four levels of addresses are used in an internet following the TCP/IP protocols: physical (link) addresses, logical (IP) addresses, port addresses, and specific addresses.
- The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN.
- The IP address uniquely defines a host on the Internet.
- The port address identifies a process on a host.
- A specific address is a user-friendly address.

2.9 PRACTICE SET

Review Questions

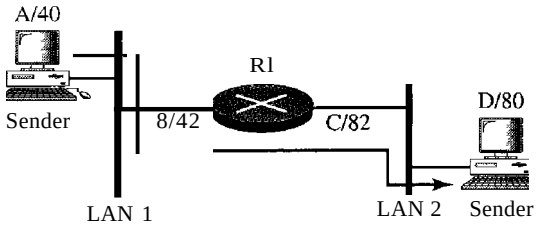
1. List the layers of the Internet model.
2. Which layers in the Internet model are the network support layers?
3. Which layer in the Internet model is the user support layer?
4. What is the difference between network layer delivery and transport layer delivery?

5. What is a peer-to-peer process?
6. How does information get passed from one layer to the next in the Internet model?
7. What are headers and trailers, and how do they get added and removed?
8. What are the concerns of the physical layer in the Internet model?
9. What are the responsibilities of the data link layer in the Internet model?
10. What are the responsibilities of the network layer in the Internet model?
11. What are the responsibilities of the transport layer in the Internet model?
12. What is the difference between a port address, a logical address, and a physical address?
13. Name some services provided by the application layer in the Internet model.
14. How do the layers of the Internet model correlate to the layers of the OSI model?

Exercises

15. How are OSI and ISO related to each other?
16. Match the following to one or more layers of the OSI model:
    a. Route determination
    b. Flow control
    c. Interface to transmission media
    d. Provides access for the end user
17. Match the following to one or more layers of the OSI model:
    a. Reliable process-to-process message delivery
    b. Route selection
    c. Defines frames
    d. Provides user services such as e-mail and file transfer
    e. Transmission of bit stream across physical medium
18. Match the following to one or more layers of the OSl model:
    a. Communicates directly with user's application program
    b. Error correction and retransmission
    c. Mechanical, electrical, and functional interface
    d. Responsibility for carrying frames between adjacent nodes
19. Match the following to one or more layers of the OSI model:
    a. Format and code conversion services
    b. Establishes, manages, and terminates sessions
    c. Ensures reliable transmission of data
    d. Log-in and log-out procedures
    e. Provides independence from differences in data representation
20. In Figure 2.22, computer A sends a message to computer D via LANl, router Rl, and LAN2. Show the contents of the packets and frames at the network and data link layer for each hop interface.

Figure 2.22 *Exercise 20*



21. In Figure 2.22, assume that the communication is between a process running at computer A with port address *i* and a process running at computer D with port address *j*. Show the contents of packets and frames at the network, data link, and transport layer for each hop.
22. Suppose a computer sends a frame to another computer on a bus topology LAN. The physical destination address of the frame is corrupted during the transmission. What happens to the frame? How can the sender be informed about the situation?
23. Suppose a computer sends a packet at the network layer to another computer somewhere in the Internet. The logical destination address of the packet is corrupted. What happens to the packet? How can the source computer be informed of the situation?
24. Suppose a computer sends a packet at the transport layer to another computer somewhere in the Internet. There is no process with the destination port address running at the destination computer. What will happen?
25. If the data link layer can detect errors between hops, why do you think we need another checking mechanism at the transport layer?

Research Activities

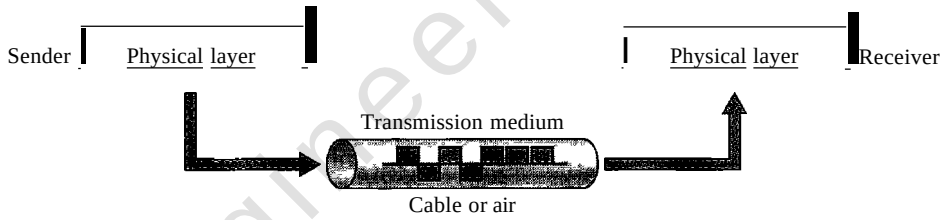
26. Give some advantages and disadvantages of combining the session, presentation, and application layer in the OSI model into one single application layer in the Internet model.
27. Dialog control and synchronization are two responsibilities of the session layer in the OSI model. Which layer do you think is responsible for these duties in the Internet model? Explain your answer.
28. Translation, encryption, and compression are some of the duties of the presentation layer in the OSI model. Which layer do you think is responsible for these duties in the Internet model? Explain your answer.
29. There are several transport layer models proposed in the OSI model. Find all of them. Explain the differences between them.
30. There are several network layer models proposed in the OSI model. Find all of them. Explain the differences between them.

CHAPTER 7

Transmission Media

We discussed many issues related to the physical layer in Chapters 3 through 6. In this chapter, we discuss transmission media. Transmission media are actually located below the physical layer and are directly controlled by the physical layer. You could say that transmission media belong to layer zero. Figure 7.1 shows the position of transmission media in relation to the physical layer.

Figure 7.1 *Transmission medium and physical layer*



A transmission **medium** can be broadly defined as anything that can carry information from a source to a destination. For example, the transmission medium for two people having a dinner conversation is the air. The air can also be used to convey the message in a smoke signal or semaphore. For a written message, the transmission medium might be a mail carrier, a truck, or an airplane.

In data communications the definition of the information and the transmission medium is more specific. The transmission medium is usually free space, metallic cable, or fiber-optic cable. The information is usually a signal that is the result of a conversion of data from another form.

The use of long-distance communication using electric signals started with the invention of the telegraph by Morse in the 19th century. Communication by telegraph was slow and dependent on a metallic medium.

Extending the range of the human voice became possible when the telephone was invented in 1869. Telephone communication at that time also needed a metallic medium to carry the electric signals that were the result of a conversion from the human voice.

The communication was, however, unreliable due to the poor quality of the wires. The lines were often noisy and the technology was unsophisticated.

Wireless communication started in 1895 when Hertz was able to send high-frequency signals. Later, Marconi devised a method to send telegraph-type messages over the Atlantic Ocean.

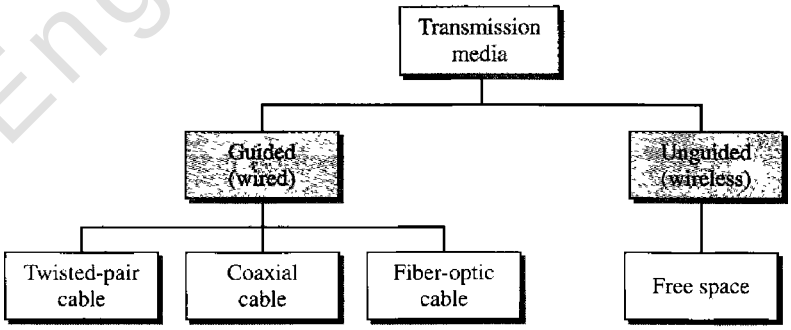
We have come a long way. Better metallic media have been invented (twisted-pair and coaxial cables, for example). The use of optical fibers has increased the data rate incredibly. Free space (air, vacuum, and water) is used more efficiently, in part due to the technologies (such as modulation and multiplexing) discussed in the previous chapters.

As discussed in Chapter 3, computers and other telecommunication devices use signals to represent data. These signals are transmitted from one device to another in the form of electromagnetic energy, which is propagated through transmission media.

Electromagnetic energy, a combination of electric and magnetic fields vibrating in relation to each other, includes power, radio waves, infrared light, visible light, ultraviolet light, and X, gamma, and cosmic rays. Each of these constitutes a portion of the electromagnetic spectrum. Not all portions of the spectrum are currently usable for telecommunications, however. The media to harness those that are usable are also limited to a few types.

In telecommunications, transmission media can be divided into two broad categories: guided and unguided. Guided media include twisted-pair cable, coaxial cable, and fiber-optic cable. Unguided medium is free space. Figure 7.2 shows this taxonomy.

Figure 7.2 *Classes of transmission media*



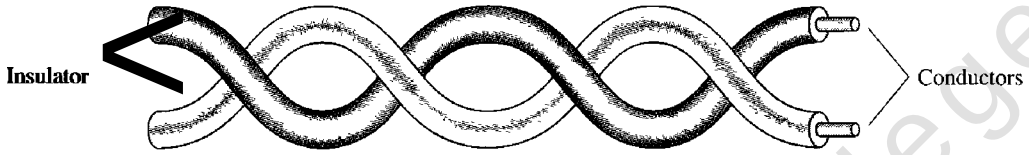
7.1 GUIDED MEDIA

Guided media, which are those that provide a conduit from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light.

Twisted-Pair Cable

A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together, as shown in Figure 7.3.

Figure 7.3 *Twisted-pair cable*



One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two.

In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals.

If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e,g., one is closer and the other is farther). This results in a difference at the receiver. By twist,ing the pairs, a balance is maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true. Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals. The unwanted signals are mostly canceled out. From the above discussion, it is clear that the number of twists per unit of length (e.g., inch) has some effect on the quality of the cable.

Unshielded Versus Shielded Twisted-Pair Cable

The most common twisted-pair cable used in communications is referred to as unshielded twisted-pair (UTP). IBM has also produced a version of twisted-pair cable for its use called shielded twisted-pair (STP). STP cable has a metal foil or braided-mesh covering that encases each pair of insulated conductors. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive. Figure 7.4 shows the difference between UTP and STP. Our discussion focuses primarily on UTP because STP is seldom used outside of IBM.

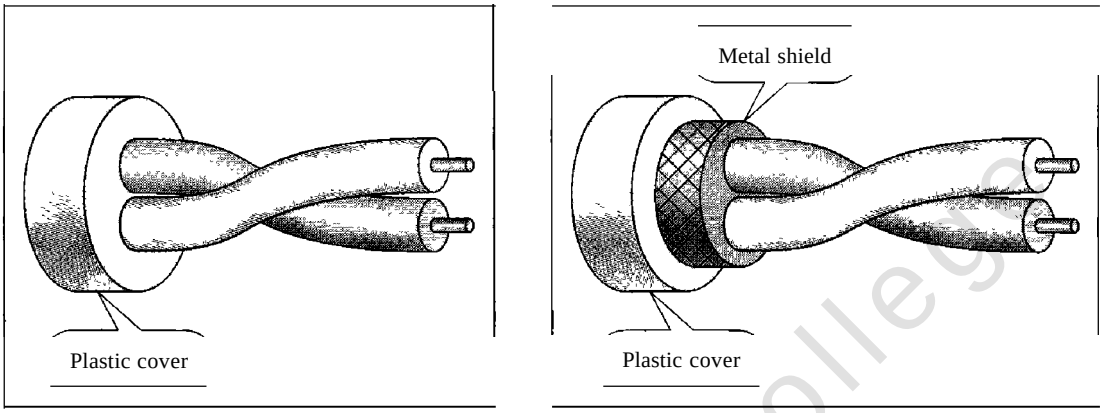
Categories

The Electronic Industries Association (EIA) has developed standards to classify unshielded twisted-pair cable into seven categories. Categories are determined by cable quality, with 1 as the lowest and 7 as the highest. Each EIA category is suitable for specific uses. Table 7.I shows these categories.

Connectors

The most common UTP connector is RJ45 (RJ stands for registered jack), as shown in Figure 7.5. The RJ45 is a keyed connector, meaning the connector can be inserted in only one way.

Figure 7.4 *UTP and STP cables*



a.UTP b.STP

Table 7.1 *Categories of unshielded twisted-pair cables*

Category	*Specification*	*Data Rate (Mbps)*	*Use*
I	Unshielded twisted-pair used in telephone	< 0.1	Telephone
2	Unshielded twisted-pair originally used in T-lines	2	T-llines
3	Improved CAT 2 used in LANs	10	LANs
4	Improved CAT 3 used in Token Ring networks	20	LANs
5	Cable wire is normally 24 AWG with a jacket and outside sheath	100	LANs
SE	An extension to category 5 that includes extra features to minimize the crosstalk and electromagnetic interference	125	LANs
6	A new category with matched components coming from the same manufacturer. The cable must be tested at a 200-Mbps data rate.	200	LANs
7	Sometimes called SSTP (shielded screen twisted-pair). Each pair is individually wrapped in a helical metallic foil followed by a metallic foil shield in addition to the outside sheath. The shield decreases the effect of crosstalk: and increases the data rate.	600	LANs

Performance

One way to measure the performance of twisted-pair cable is to compare attenuation versus frequency and distance. A twisted-pair cable can pass a wide range of frequencies. However, Figure 7.6 shows that with increasing frequency, the attenuation, measured in decibels per kilometer (dB/km), sharply increases with frequencies above 100 kHz. Note that *gauge* is a measure of the thickness of the wire.

Figure 7.5 *UTP connector*

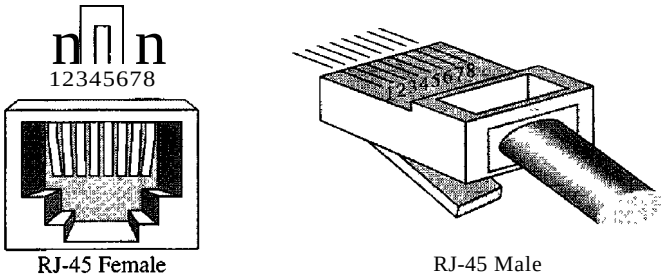
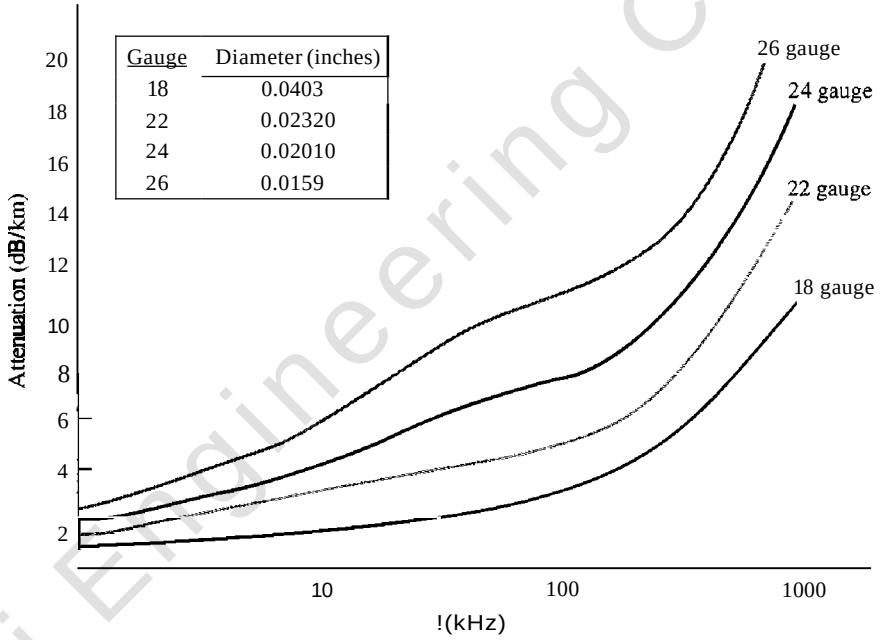


Figure 7.6 *UTP performance*



Applications

Twisted-pair cables are used in telephone lines to provide voice and data channels. The local loop-the line that connects subscribers to the central telephone office---commonly consists of unshielded twisted-pair cables. We discuss telephone networks in Chapter 9.

The DSL lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables. We discuss DSL technology in Chapter 9.

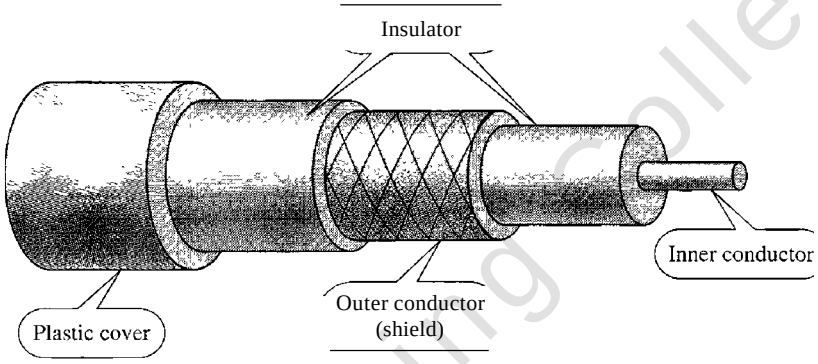
Local-area networks, such as lOBase-T and lOOBase-T, also use twisted-pair cables. We discuss these networks in Chapter 13.

Coaxial Cable

Coaxial cable (or *coax)* carries signals of higher frequency ranges than those in twisted-pair cable, in part because the two media are constructed quite differently. Instead of

having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover (see Figure 7.7).

Figure 7.7 *Coaxial cable*



Coaxial Cable Standards

Coaxial cables are categorized by their radio government (RG) ratings. Each RG number denotes a unique set of physical specifications, including the wire gauge of the inner conductor, the thickness and type of the inner insulator, the construction of the shield, and the size and type of the outer casing. Each cable defined by an RG rating is adapted for a specialized function, as shown in Table 7.2.

Table 7.2 *Categories of coaxial cables*

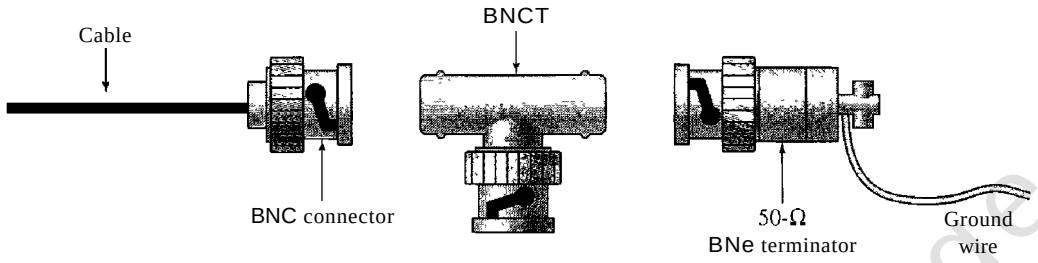
Category	*Impedance*	*Use*
RG-59	75 Ω	Cable TV
RG-58	50 Ω	Thin Ethernet
RG-ll	50 Ω	Thick Ethernet

Coaxial Cable Connectors

To connect coaxial cable to devices, we need coaxial connectors. The most common type of connector used today is the Bayone-Neill-Concelman (BNe), connector. Figure 7.8 shows three popular types of these connectors: the BNC connector, the BNC T connector, and the BNC terminator.

The BNC connector is used to connect the end of the cable to a device, such as a TV set. The BNC T connector is used in Ethernet networks (see Chapter 13) to branch out to a connection to a computer or other device. The BNC terminator is used at the end of the cable to prevent the reflection of the signal.

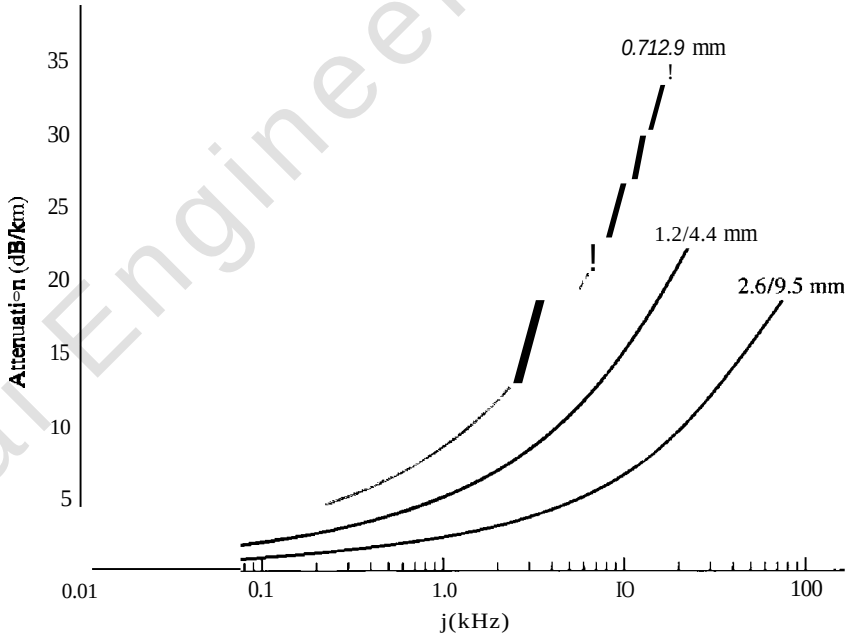
Figure 7.8 *BNC connectors*



Performance

As we did with twisted-pair cables, we can measure the performance of a coaxial cable. We notice in Figure 7.9 that the attenuation is much higher in coaxial cables than in twisted-pair cable. In other words, although coaxial cable has a much higher bandwidth, the signal weakens rapidly and requires the frequent use of repeaters.

Figure 7.9 *Coaxial cable peiformance*



Applications

Coaxial cable was widely used in analog telephone networks where a single coaxial network could carry 10,000 voice signals. Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps. However, coaxial cable in telephone networks has largely been replaced today with fiber-optic cable.

Cable TV networks (see Chapter 9) also use coaxial cables. In the traditional cable TV network, the entire network used coaxial cable. Later, however, cable TV providers

replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable.

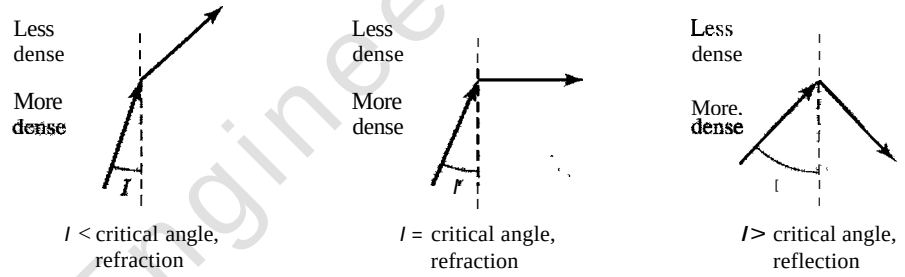
Another common application of coaxial cable is in traditional Ethernet LANs (see Chapter 13). Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs. The 10Base-2, or Thin Ethernet, uses RG-58 coaxial cable with BNe connectors to transmit data at 10 Mbps with a range of 185 m. The lOBase5, or Thick Ethernet, uses RG-11 (thick coaxial cable) to transmit 10 Mbps with a range of 5000 m. Thick Ethernet has specialized connectors.

Fiber-Optic Cable

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light.

Light travels in a straight line as long as it is moving through a single uniform substance. If a ray of light traveling through one substance suddenly enters another substance (of a different density), the ray changes direction. Figure 7.10 shows how a ray of light changes direction when going from a more dense to a less dense substance.

Figure 7.10 *Bending of light ray*



As the figure shows, if the angle of incidence I (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the critical angle, the ray refracts and moves closer to the surface. If the angle of incidence is equal to the critical angle, the light bends along the interface. If the angle is greater than the critical angle, the ray reflects (makes a turn) and travels again in the denser substance. Note that the critical angle is a property of the substance, and its value differs from one substance to another.

Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. See Figure 7.11.

Propagation Modes

Current technology supports two modes (multimode and single mode) for propagating light along optical channels, each requiring fiber with different physical characteristics. Multimode can be implemented in two forms: step-index or graded-index (see Figure 7.12).

Figure 7.11 *Opticaljiber*

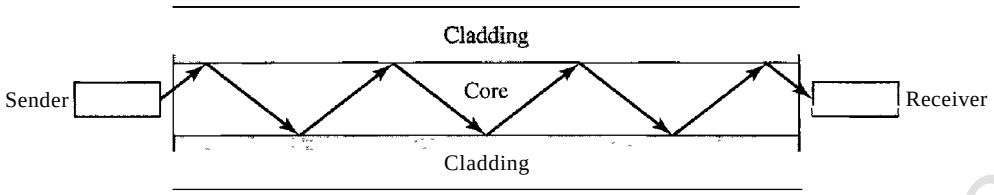
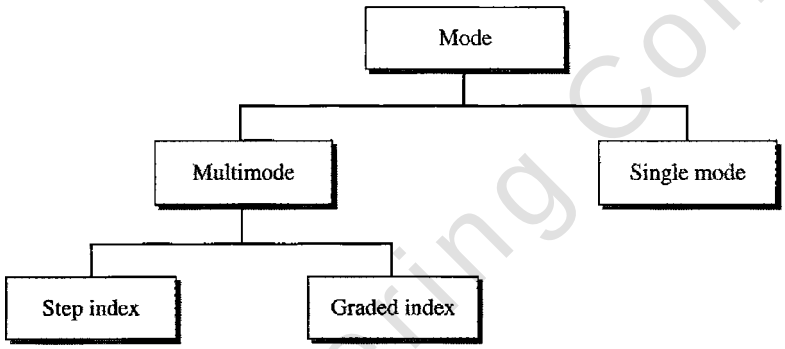


Figure 7.12 *Propagation modes*



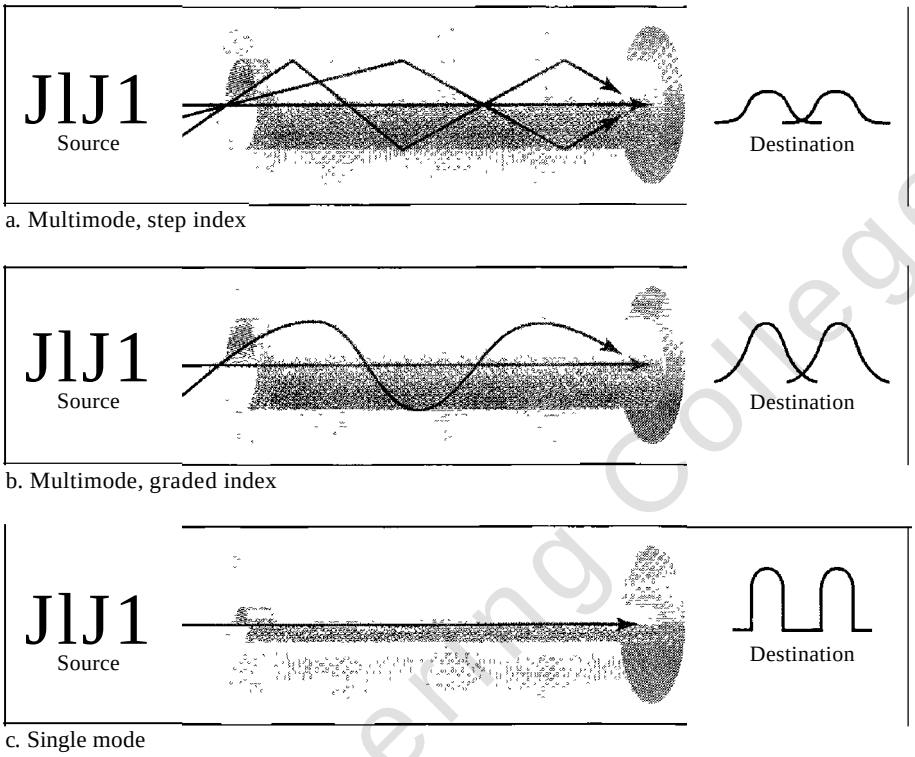
Multimode Multimode is so named because multiple beams from a light source move through the core in different paths. How these beams move within the cable depends on the structure ofthe core, as shown in Figure 7.13.

In multimode step-index fiber, the density of the core remains constant from the center to the edges. A beam of light moves through this constant density in a straight line until it reaches the interface of the core and the cladding. At the interface, there is an abrupt change due to a lower density; this alters the angle of the beam's motion. The term *step index* refers to the suddenness of this change, which contributes to the distortion of the signal as it passes through the fiber.

A second type of fiber, called multimode graded-index fiber, decreases this distortion of the signal through the cable. The word *index* here refers to the index of refraction. As we saw above, the index of refraction is related to density. A graded-index fiber, therefore, is one with varying densities. Density is highest at the center of the core and decreases gradually to its lowest at the edge. Figure 7.13 shows the impact of this variable density on the propagation of light beams.

Single-Mode Single-mode uses step-index fiber and a highly focused source of light that limits beams to a small range of angles, all close to the horizontal. The single-mode fiber itself is manufactured with a much smaller diameter than that of multimode fiber, and with substantiallY lower density (index of refraction). The decrease in density results in a critical angle that is close enough to 90° to make the propagation of beams almost horizontal. In this case, propagation of different beams is almost identical, and delays are negligible. All the beams arrive at the destination "together" and can be recombined with little distortion to the signal (see Figure 7.13).

Figure 7.13 *Modes*



Fiber Sizes

Optical fibers are defined by the ratio of the diameter of their core to the diameter of their cladding, both expressed in micrometers. The common sizes are shown in Table 7.3. Note that the last size listed is for single-mode only.

Table 7.3 *Fiber types*

Type	*Core (μm)*	*Cladding (μm)*	*Mode*
501125	50.0	125	Multimode, graded index
62.51125	62.5	125	Multimode, graded index
100/125	100.0	125	Multimode, graded index
7/125	7.0	125	Single mode

Cable Composition

Figure 7.14 shows the composition of a typical fiber-optic cable. The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable. Kevlar is a strong material used in the fabrication of bulletproof vests. Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.

Fiber-Optic Cable Connectors

There are three types of connectors for fiber-optic cables, as shown in Figure 7.15.

Figure 7.14 *Fiber construction*

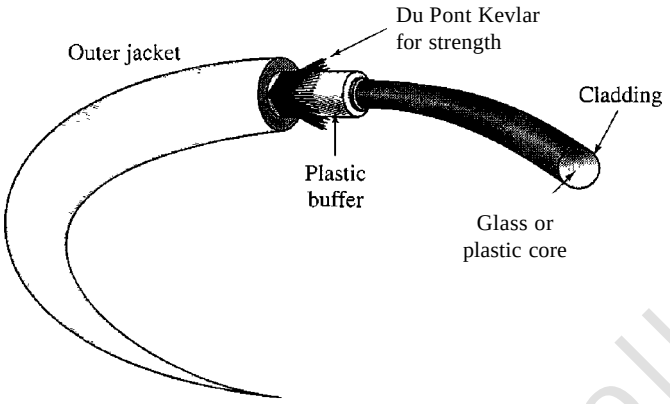
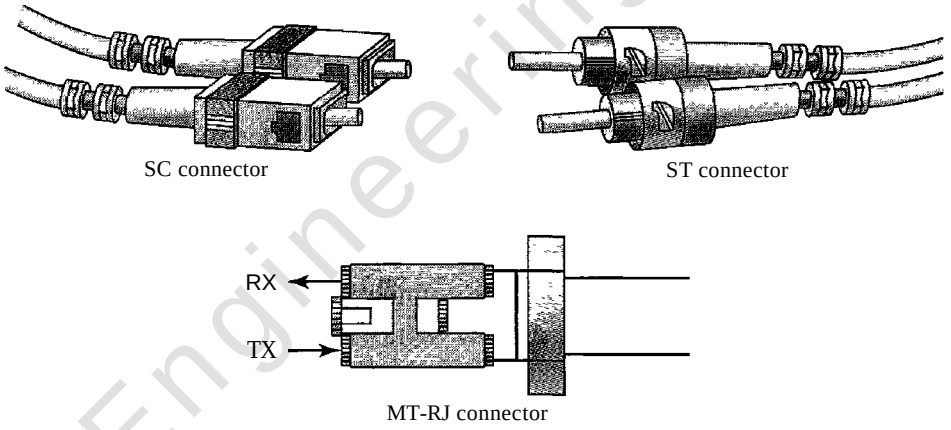


Figure 7.15 *Fiber-optic cable connectors*



The **subscriber channel** (SC) **connector** is used for cable TV. It uses a push/pull locking system. The **straight-tip** (ST) **connector** is used for connecting cable to networking devices. It uses a bayonet locking system and is more reliable than SC. **MT-RJ** is a connector that is the same size as RJ45.

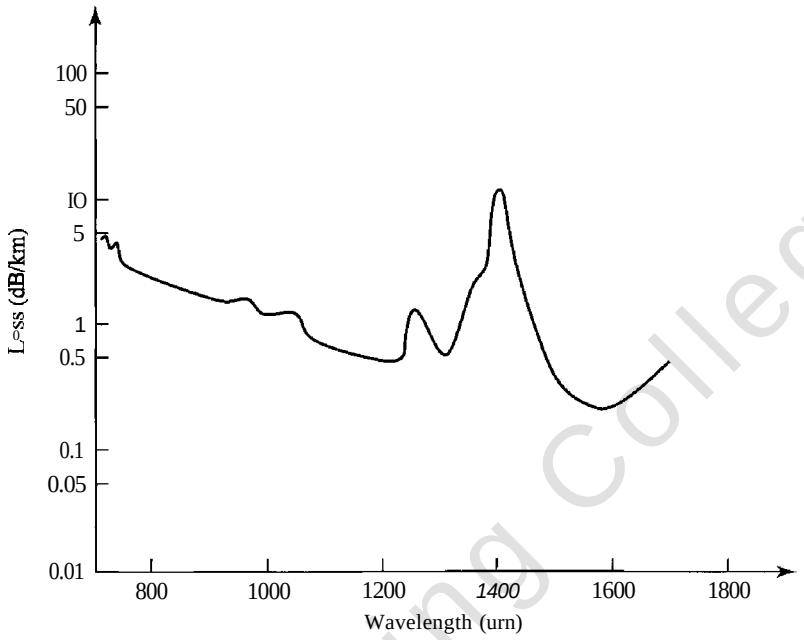
Performance

The plot of attenuation versus wavelength in Figure 7.16 shows a very interesting phenomenon in fiber-optic cable. Attenuation is flatter than in the case of twisted-pair cable and coaxial cable. The performance is such that we need fewer (actually 10 times less) repeaters when we use fiber-optic cable.

Applications

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective. Today, with wavelength-division multiplexing (WDM), we can transfer

Figure 7.16 *Optical fiber performance*



data at a rate of 1600 Gbps. The SONET network that we discuss in Chapter 17 provides such a backbone.

Some cable TV companies use a combination of optical fiber and coaxial cable, thus creating a hybrid network. Optical fiber provides the backbone structure while coaxial cable provides the connection to the user premises. This is a cost-effective configuration since the narrow bandwidth requirement at the user end does not justify the use of optical fiber.

Local-area networks such as 100Base-FX network (Fast Ethernet) and 1000Base-X also use fiber-optic cable.

Advantages and Disadvantages of Optical Fiber

Advantages Fiber-optic cable has several advantages over metallic cable (twisted-pair or coaxial).

- Higher bandwidth. Fiber-optic cable can support dramatically higher bandwidths (and hence data rates) than either twisted-pair or coaxial cable. Currently, data rates and bandwidth utilization over fiber-optic cable are limited not by the medium but by the signal generation and reception technology available.
- Less signal attenuation. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.
- Immunity to electromagnetic interference. Electromagnetic noise cannot affect fiber-optic cables.
- Resistance to corrosive materials. Glass is more resistant to corrosive materials than copper.

- Light weight. Fiber-optic cables are much lighter than copper cables.
- Greater immunity to tapping. Fiber-optic cables are more immune to tapping than copper cables. Copper cables create antenna effects that can easily be tapped.

Disadvantages There are some disadvantages in the use of optical fiber.

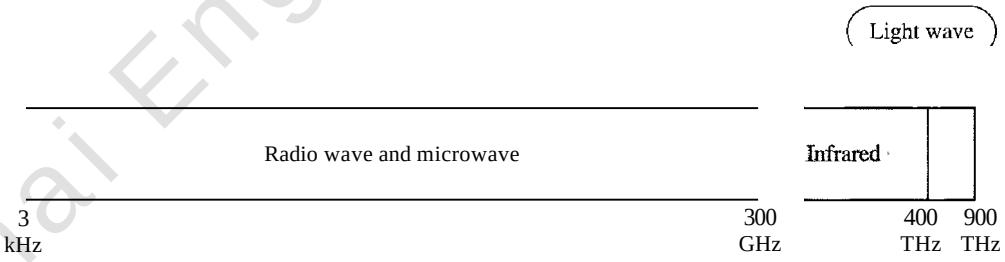
- Installation and maintenance. Fiber-optic cable is a relatively new technology. Its installation and maintenance require expertise that is not yet available everywhere.
- Unidirectional light propagation. Propagation of light is unidirectional. If we need bidirectional communication, two fibers are needed.
- Cost. The cable and the interfaces are relatively more expensive than those of other guided media. If the demand for bandwidth is not high, often the use of optical fiber cannot be justified.

7.2 UNGUIDED MEDIA: WIRELESS

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication. Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them.

Figure 7.17 shows the part of the electromagnetic spectrum, ranging from 3 kHz to 900 THz, used for wireless communication.

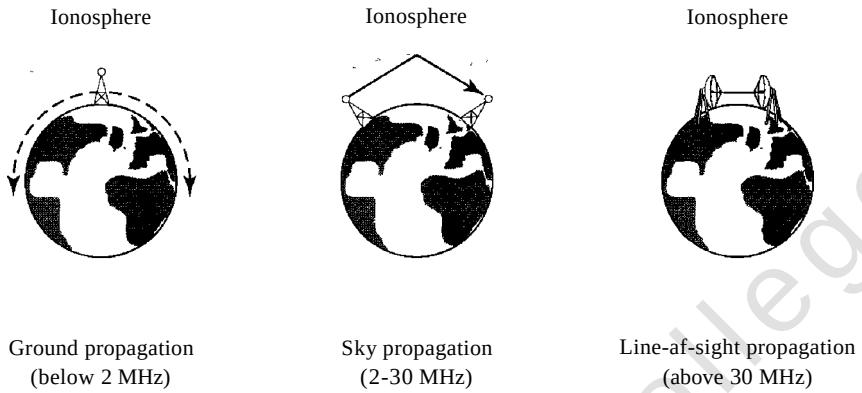
Figure 7.17 *Electromagnetic spectrumfor wireless communication*



Unguided signals can travel from the source to destination in several ways: ground propagation, sky propagation, and line-of-sight propagation, as shown in Figure 7.18.

In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance. In sky propagation, higher-frequency radio waves radiate upward into the ionosphere (the layer of atmosphere where particles exist as ions) where they are reflected back to earth. This type of transmission allows for greater distances with lower output power. In line-or-sight propagation, very high-frequency signals are transmitted in straight lines directly from antenna to antenna. Antennas must be directional, facing each other,

Figure 7.18 *Propagation methods*



and either tall enough or close enough together not to be affected by the curvature of the earth. Line-of-sight propagation is tricky because radio transmissions cannot be completely focused.

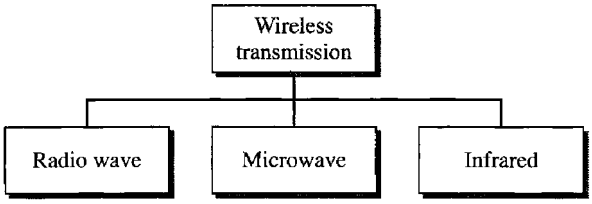
The section of the electromagnetic spectrum defined as radio waves and microwaves is divided into eight ranges, called *bands*, each regulated by government authorities. These bands are rated from *very low frequency* (VLF) to *extremely high frequency* (EHF). Table 7.4 lists these bands, their ranges, propagation methods, and some applications.

Table 7.4 *Bands*

Band	*Range*	*Propagation*	*Application*
VLF (very low frequency)	3-30 kHz	Ground	Long-range radio navigation
LF (low frequency)	30-300 kHz	Ground	Radio beacons and navigational locators
MF (middle frequency)	300 kHz-3 MHz	Sky	AM radio
HF (high frequency)	3-30 MHz	Sky	Citizens band (CB), ship-to-aircraft communication
VHF (very high frequency)	30-300 MHz	Sky and line-of-sight	VHF TV, FM radio
UHF (ultrahigh frequency)	300 MHz-3 GHz	Line-of-sight	UHF TV, cellular phones, paging, satellite
SHF (superhigh frequency)	3-30 GHz	Line-of-sight	Satellite communication
EHF (extremely high frequency)	30-300 GHz	Line-of-sight	Radar, satellite

We can divide wireless transmission into three broad groups: radio waves, microwaves, and infrared waves. See Figure 7.19.

Figure 7.19 *Wireless transmission waves*



Radio Waves

Although there is no clear-cut demarcation between radio waves and microwaves, electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves; waves ranging in frequencies between 1 and 300 GHz are called microwaves. However, the behavior of the waves, rather than the frequencies, is a better criterion for classification.

Radio waves, for the most part, are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna. The omnidirectional property has a disadvantage, too. The radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band.

Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio.

Radio waves, particularly those of low and medium frequencies, can penetrate walls. This characteristic can be both an advantage and a disadvantage. It is an advantage because, for example, an AM radio can receive signals inside a building. It is a disadvantage because we cannot isolate a communication to just inside or outside a building. The radio wave band is relatively narrow, just under 1 GHz, compared to the microwave band. When this band is divided into subbands, the subbands are also narrow, leading to a low data rate for digital communications.

Almost the entire band is regulated by authorities (e.g., the FCC in the United States). Using any part of the band requires permission from the authorities.

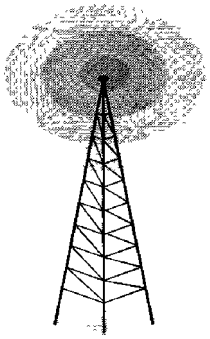
Omnidirectional Antenna

Radio waves use omnidirectional antennas that send out signals in all directions. Based on the wavelength, strength, and the purpose of transmission, we can have several types of antennas. Figure 7.20 shows an omnidirectional antenna.

Applications

The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. AM and FM radio, television, maritime radio, cordless phones, and paging are examples of multicasting.

Figure 7.20 *Omnidirectional antenna*



Radio waves are used for multicast communications, such as radio and television, and paging systems.

Microwaves

Electromagnetic waves having frequencies between I and 300 GHz are called microwaves.

Microwaves are unidirectional. When an antenna transmits microwave waves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas. The following describes some characteristics of microwave propagation:

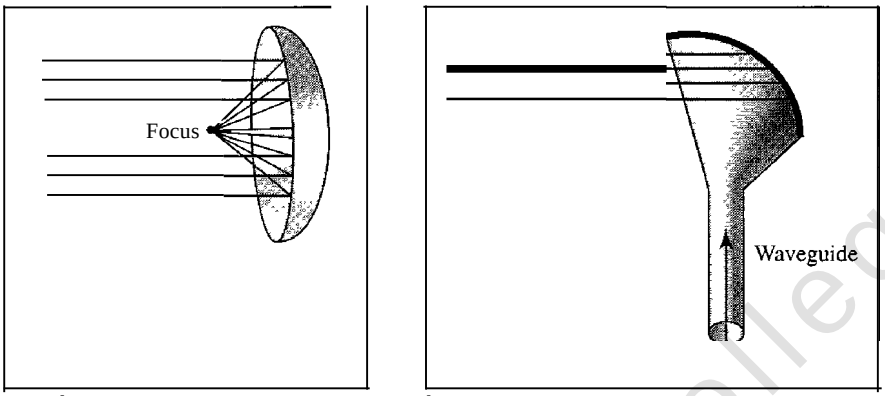
- Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles do not allow two short towers to communicate by using microwaves. Repeaters are often needed for long-distance communication.
- Very high-frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.
- The microwave band is relatively wide, almost 299 GHz. Therefore wider subbands can be assigned, and a high data rate is possible
- Use of certain portions of the band requires permission from authorities.

Unidirectional Antenna

Microwaves need unidirectional antennas that send out signals in one direction. Two types of antennas are used for microwave communications: the parabolic dish and the hom (see Figure 7.21).

A parabolic dish antenna is based on the geometry of a parabola: Every line parallel to the line of symmetry (line of sight) reflects off the curve at angles such that all the lines intersect in a common point called the focus. The parabolic dish works as a

Figure 7.21 *Unidirectional antennas*



a. Dish antenna b. Horn antenna

funnel, catching a wide range of waves and directing them to a common point. In this way, more of the signal is recovered than would be possible with a single-point receiver.

Outgoing transmissions are broadcast through a horn aimed at the dish. The microwaves hit the dish and are deflected outward in a reversal of the receipt path.

A horn antenna looks like a gigantic scoop. Outgoing transmissions are broadcast up a stem (resembling a handle) and deflected outward in a series of narrow parallel beams by the curved head. Received transmissions are collected by the scooped shape of the horn, in a manner similar to the parabolic dish, and are deflected down into the stem.

Applications

Microwaves, due to their unidirectional properties, are very useful when unicast (one-to-one) communication is needed between the sender and the receiver. They are used in cellular phones (Chapter 16), satellite networks (Chapter 16), and wireless LANs (Chapter 14).

Microwaves are used for unicast communication such as cellular telephones, satellite networks, and wireless LANs.

Infrared

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

Applications

The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate. The *Infrared Data Association* (IrDA), an association for sponsoring the use of infrared waves, has established standards for using these signals for communication between devices such as keyboards, mice, PCs, and printers. For example, some manufacturers provide a special port called the IrDA port that allows a wireless keyboard to communicate with a PC. The standard originally defined a data rate of 75 kbps for a distance up to 8 m. The recent standard defines a data rate of 4 Mbps.

Infrared signals defined by IrDA transmit through line of sight; the IrDA port on the keyboard needs to point to the PC for transmission to occur.

Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation.

7.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Transmission media is discussed in Section 3.8 of [GW04], Chapter 4 of [Sta04], Section 2.2 and 2.3 of [Tan03]. [SSS05] gives a full coverage of transmission media.

7.4 KEY TERMS

angle of incidence
Bayone-Neil-Concelman (BNC) connector
cladding
coaxial cable
core
critical angle
electromagnetic spectrum
fiber-optic cable
gauge
ground propagation
guided media
horn antenna
infrared wave
IrDA port
line-of-sight propagation
microwave
MT-RJ
multimode graded-index fiber
multimode step-index fiber
omnidirectional antenna
optical fiber
parabolic dish antenna
Radio Government (RG) number
radio wave
reflection
refraction
RJ45

shielded twisted-pair (STP)
single-mode fiber
sky propagation
straight-tip (ST) connector
subscriber channel (SC) connector
transmission medium
twisted-pair cable
unguided medium
unidirectional antenna
unshielded twisted-pair (UTP)
wireless communication

7.5 SUMMARY

- Transmission media lie below the physical layer.
- A guided medium provides a physical conduit from one device to another. Twisted-pair cable, coaxial cable, and optical fiber are the most popular types of guided media.
- Twisted-pair cable consists of two insulated copper wires twisted together. Twisted-pair cable is used for voice and data communications.
- Coaxial cable consists of a central conductor and a shield. Coaxial cable can carry signals of higher frequency ranges than twisted-pair cable. Coaxial cable is used in cable TV networks and traditional Ethernet LANs.
- Fiber-optic cables are composed of a glass or plastic inner core surrounded by cladding, all encased in an outside jacket. Fiber-optic cables carry data signals in the form of light. The signal is propagated along the inner core by reflection. Fiber-optic transmission is becoming increasingly popular due to its noise resistance, low attenuation, and high-bandwidth capabilities. Fiber-optic cable is used in backbone networks, cable TV networks, and Fast Ethernet networks.
- Unguided media (free space) transport electromagnetic waves without the use of a physical conductor.
- Wireless data are transmitted through ground propagation, sky propagation, and line-of-sight propagation.Wireless waves can be classified as radio waves, microwaves, or infrared waves. Radio waves are omnidirectional; microwaves are unidirectional. Microwaves are used for cellular phone, satellite, and wireless LAN communications.
- Infrared waves are used for short-range communications such as those between a PC and a peripheral device. It can also be used for indoor LANs.

7.6 PRACTICE SET

Review Questions

1. What is the position of the transmission media in the OSI or the Internet model?
2. Name the two major categories of transmission media.
3. How do guided media differ from unguided media?
4. What are the three major classes of guided media?
5. What is the significance of the twisting in twisted-pair cable?

6. What is refraction? What is reflection?
7. What is the purpose of cladding in an optical fiber?
8. Name the advantages of optical fiber over twisted-pair and coaxial cable.
9. How does sky propagation differ from line-of-sight propagation?
10. What is the difference between omnidirectional waves and unidirectional waves?

Exercises

11. Using Figure 7.6, tabulate the attenuation (in dB) of a 18-gauge UTP for the indicated frequencies and distances.

Table 7.5 *Attenuation/or I8-gauge UTP*

Distance	*dB at* 1 *KHz*	*dRat 10KHz*	*dB at 100 KHz*
1 Krn			
lOKm			
15 Krn			
20Km			

12. Use the result of Exercise 11 to infer that the bandwidth of a UTP cable decreases with an increase in distance.
13. **If** the power at the beginning of a 1 KIn 18-gauge UTP is 200 mw, what is the power at the end for frequencies 1 KHz, 10 KHz, and 100 KHz? Use the result of Exercise 11.
14. Using Figure 7.9, tabulate the attenuation (in dB) of a 2.6/9.5 mm coaxial cable for the indicated frequencies and distances.

Table 7.6 *Attenuation/or* 2.6/9.5 *mm coaxial cable*

Distance	*dB at* 1 *KHz*	*dB at 10 KHz*	*dB at 100KHz*
1 Km			
lOKrn			
15Km			
20Km			

15. Use the result of Exercise 14 to infer that the bandwidth of a coaxial cable decreases with the increase in distance.
16. **If** the power at the beginning of a 1 KIn 2.6/9.5 mm coaxial cable is 200 mw, what is the power at the end for frequencies 1 KHz, 10KHz, and 100 KHz? Use the result of Exercise 14.
17. Calculate the bandwidth of the light for the following wavelength ranges (assume a propagation speed of 2 x 10^8 m):
    a. 1000 to 1200 nm
    b. 1000 to 1400 nm

18. The horizontal axes in Figure 7.6 and 7.9 represent frequencies. The horizontal axis in Figure 7.16 represents wavelength. Can you explain the reason? lfthe propagation speed **in** an optical fiber is 2 x 10^8 ill, can you change the units in the horizontal axis to frequency? Should the vertical-axis units be changed too? Should the curve be changed too?
19. Using Figure 7.16, tabulate the attenuation (in dB) of an optical fiber for the indicated wavelength and distances.

Table 7.7 *Attenuation for optical fiber*

Distance	*dB at 800 nm*	*dB at 1000 nm*	*dB at 1200nm*
1 Km			
lOKm			
15Km			
20Km			

20. A light signal is travelling through a fiber. What is the delay **in** the signal if the length of the fiber-optic cable is 10 m, 100 m, and 1 Km (assume a propagation speed of 2 x 10^8 ill)?
21. A beam oflight moves from one medium to another medium with less density. The critical angle is 60°. Do we have refraction or reflection for each of the following incident angles? Show the bending of the light ray in each case.
    a. 40°
    b. 60°
    c. 80°

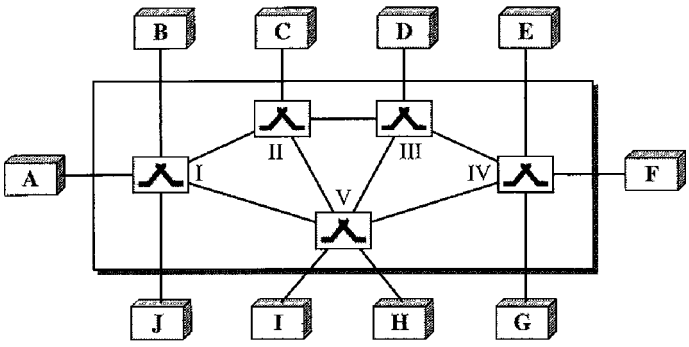
CHAPTER 8

Switching

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). These methods, however, are impractical and wasteful when applied to very large networks. The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most of the time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment.

A better solution is switching. A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure 8.1 shows a switched network.

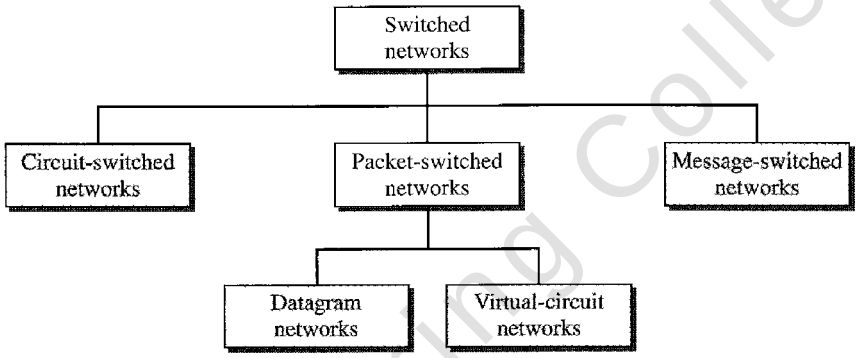
Figure 8.1 *Switched network*



The end systems (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

Traditionally, three methods of switching have been important: circuit switching, packet switching, and message switching. The first two are commonly used today. The third has been phased out in general communications but still has networking applications. We can then divide today's networks into three broad categories: circuit-switched networks, packet-switched networks, and message-switched. Packet-switched networks can funher be divided into two subcategories-virtual-circuit networks and datagram networks-as shown in Figure 8.2.

Figure 8.2 *Taxonomy of switched networks*



We can say that the virtual-circuit networks have some common characteristics with circuit-switched and datagram networks. Thus, we first discuss circuit-switched networks, then datagram networks, and finally virtual-circuit networks.

Today the tendency in packet switching is to combine datagram networks and virtual-circuit networks. Networks route the first packet based on the datagram addressing idea, but then create a virtual-circuit network for the rest of the packets coming from the same source and going to the same destination. We will see some of these networks in future chapters.

In message switching, each switch stores the whole message and forwards it to the next switch. Although, we don't see message switching at lower layers, it is still used in some applications like electronic mail (e-mail). We will not discuss this topic in this book.

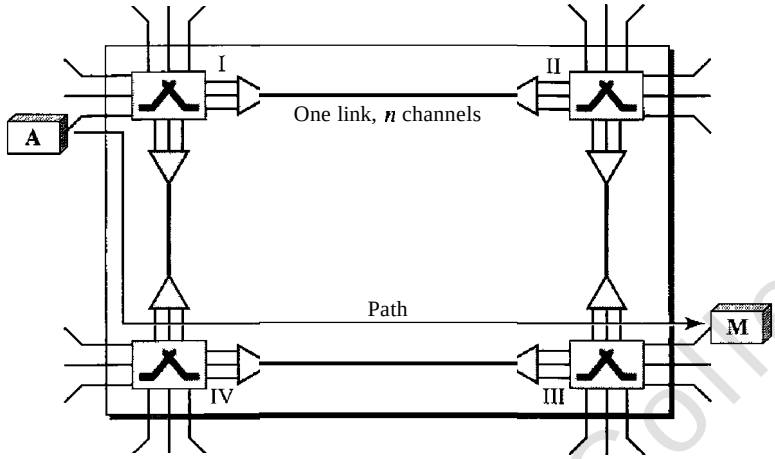
8.1 CIRCUIT-SWITCHED NETWORKS

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link. Each link is normally divided into *n* channels by using FDM or TDM as discussed in Chapter 6.

A circuit-switched network is made of a set of switches connected by physical links, in which each link is divided into *n* channels.

Figure 8.3 shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n (n* is 3 in the figure) channels by using FDM or TDM.

Figure 8.3 *A trivial circuit-switched network*



We have explicitly shown the multiplexing symbols to emphasize the division of the link into channels even though multiplexing can be implicitly included in the switch fabric.

The end systems, such as computers or telephones, are directly connected to a switch. We have shown only two end systems for simplicity. When end system A needs to communicate with end system M, system A needs to request a connection to M that must be accepted by all switches as well as by M itself. This is called the setup phase; a circuit (channel) is reserved on each link, and the combination of circuits or channels defines the dedicated path. After the dedicated path made of connected circuits (channels) is established, data transfer can take place. After all data have been transferred, the circuits are tom down.

We need to emphasize several points here:

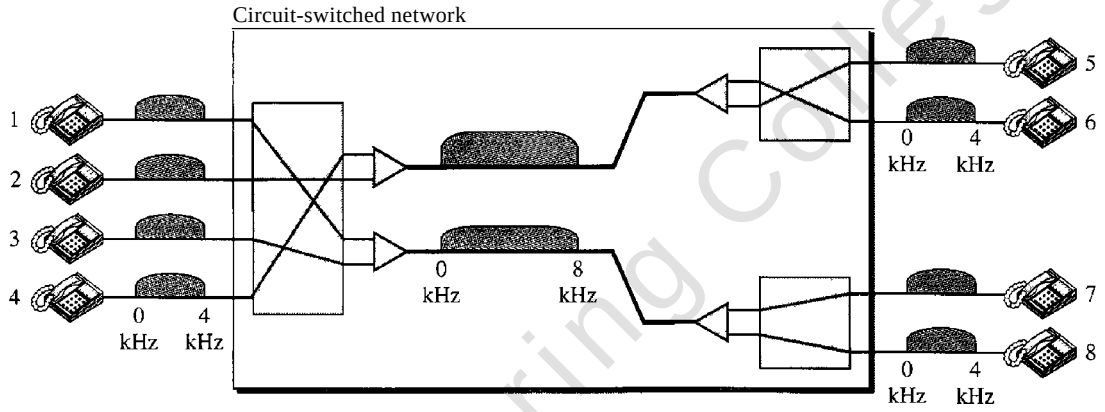
- Circuit switching takes place at the physical layer.
- Before starting communication, the stations must make a reservation for the resources to be used during the communication. These resources, such as channels (bandwidth in FDM and time slots in TDM), switch buffers, switch processing time, and switch input/output ports, must remain dedicated during the entire duration of data transfer until the teardown phase.
- Data transferred between the two stations are not packetized (physical layer transfer of the signal). The data are a continuous flow sent by the source station and received by the destination station, although there may be periods of silence.
- There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to-end addressing used during the setup phase, as we will see shortly.

In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer until the teardown phase.

Example 8.1

As a trivial example, let us use a circuit-switched network to connect eight telephones in a small area. Communication is through 4-kHz voice channels. We assume that each link uses FDM to connect a maximum of two voice channels. The bandwidth of each link is then 8 kHz. Figure 8.4 shows the situation. Telephone 1 is connected to telephone 7; 2 to 5; 3 to 8; and 4 to 6. Of course the situation may change when new connections are made. The switch controls the connections.

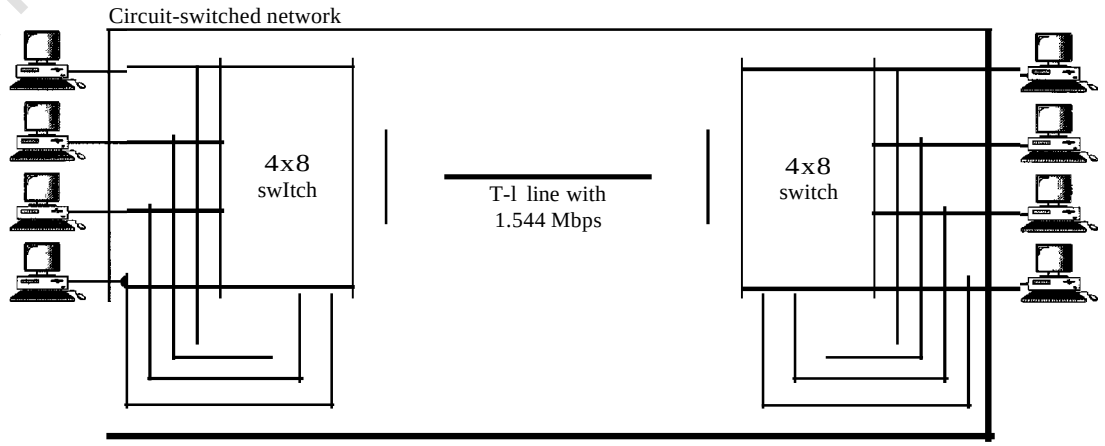
Figure 8.4 *Circuit-switched network used in Example 8.1*



Example 8.2

As another example, consider a circuit-switched network that connects computers in two remote offices of a private company. The offices are connected using a T-l line leased from a communication service provider. There are two 4 X 8 (4 inputs and 8 outputs) switches in this network. For each switch, four output ports are folded into the input ports to allow communication between computers in the same office. Four other output ports allow communication between the two offices. Figure 8.5 shows the situation.

Figure 8.5 *Circuit-switched network used* in *Example 8.2*



Three Phases

The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Setup Phase

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches. For example, in Figure 8.3, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established.

Note that end-to-end addressing is required for creating a connection between the two end systems. These can be, for example, the addresses of the computers assigned by the administrator in a TDM network, or telephone numbers in an FDM network.

Data Transfer Phase

After the establishment of the dedicated circuit (channels), the two parties can transfer data.

Teardown Phase

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

Efficiency

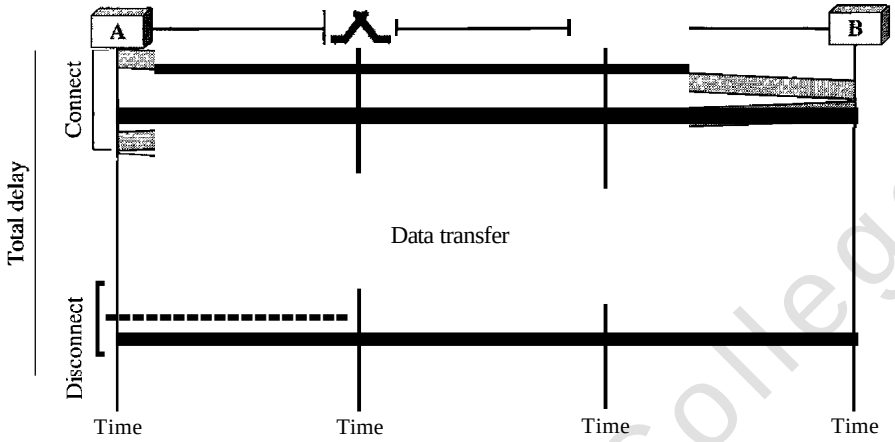
It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation. However, in computer networks, a computer can be connected to another computer even if there is no activity for a long time. In this case, allowing resources to be dedicated means that other connections are deprived.

Delay

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. Figure 8.6 shows the idea of delay in a circuit-switched network when only two switches are involved.

As Figure 8.6 shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit. The

Figure 8.6 *Delay in a circuit-switched network*



delay caused by the setup is the sum of four parts: the propagation time of the source computer request (slope of the first gray box), the request signal transfer time (height of the first gray box), the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and the signal transfer time of the acknowledgment (height of the second gray box). The delay due to data transfer is the sum of two parts: the propagation time (slope of the colored box) and data transfer time (height of the colored box), which can be very long. The third box shows the time needed to tear down the circuit. We have shown the case in which the receiver requests disconnection, which creates the maximum delay.

Circuit-Switched Technology in Telephone Networks

As we will see in Chapter 9, the telephone companies have previously chosen the circuit-switched approach to switching in the physical layer; today the tendency is moving toward other switching techniques. For example, the telephone number is used as the global address, and a signaling system (called SS7) is used for the setup and teardown phases.

Switching at the physical layer **in the traditional telephone network** uses **the circuit-switching approach.**

8.2 DATAGRAM NETWORKS

In data communications, we need to send messages from one end system to another. If the message is going to pass through a packet-switched network, it needs to be divided into packets of fixed or variable size. The size of the packet is determined by the network and the governing protocol.

In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time

for each packet. Resources are allocated on demand. The allocation is done on a first-come, first-served basis. When a switch receives a packet, no matter what is the source or destination, the packet must wait if there are other packets being processed. As with other systems in our daily life, this lack of reservation may create delay. For example, if we do not have a reservation at a restaurant, we might have to wait.

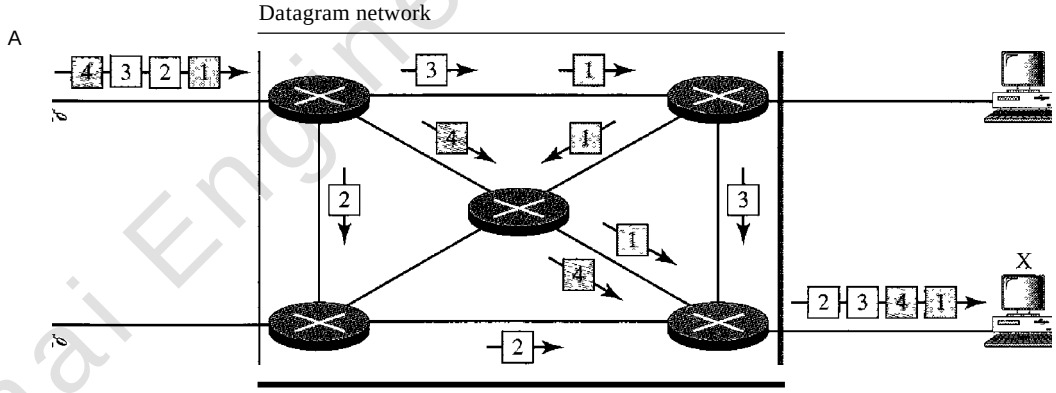
In a packet-switched network, there is no resource reservation; resources are allocated on demand.

In a datagram network, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as datagrams.

Datagram switching is normally done at the network layer. We briefly discuss datagram networks here as a comparison with circuit-switched and virtual-circuit-switched networks. In Part 4 of this text, we go into greater detail.

Figure 8.7 shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure.

Figure 8.7 *A datagram network with four switches (routers)*



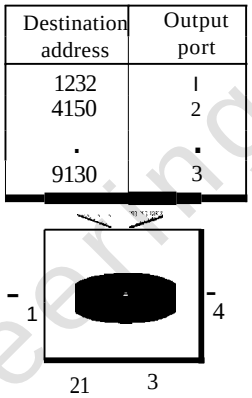
In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X. This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between the packets. Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upper-layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application.

The datagram networks are sometimes referred to as connectionless networks. The term *connectionless* here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. This is different from the table of a circuit-switched network in which each entry is created when the setup phase is completed and deleted when the teardown phase is over. Figure 8.8 shows the routing table for a switch.

Figure 8.8 *Routing table in a datagram network*



A switch in a datagram network uses a routing table that is based on the destination address.

Destination Address

Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded. This address, unlike the address in a virtual-circuit-switched network, remains the same during the entire journey of the packet.

The destination address in the header of a packet in a datagram network remains the same during the entire journey of the packet.

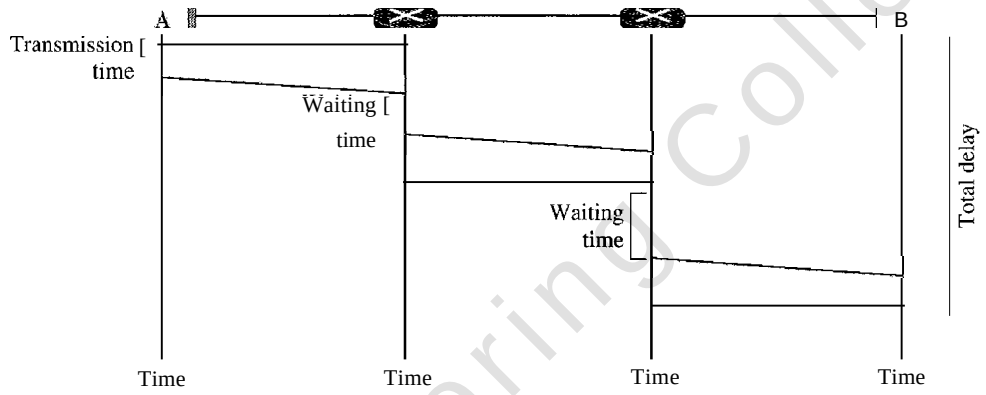
Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these minutes for other packets from other sources.

Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message. Figure 8.9 gives an example of delay in a datagram network for one single packet.

Figure 8.9 *Delay in a datagram network*



The packet travels through two switches. There are three transmission times *(3T),* three propagation delays (slopes 3τ of the lines), and two waiting times (WI + *w2)'* We ignore the processing time in each switch. The total delay is

$$\text{Total delay} = 3T + 3\tau + \text{WI} + \text{W2}$$

Datagram Networks in the Internet

As we will see in future chapters, the Internet has chosen the datagram approach to switching at the network layer. It uses the universal addresses defined in the network layer to route packets from the source to the destination.

> Switching in the Internet is done by using the datagram approach to packet switching at the network layer.

8.3 VIRTUAL-CIRCUIT NETWORKS

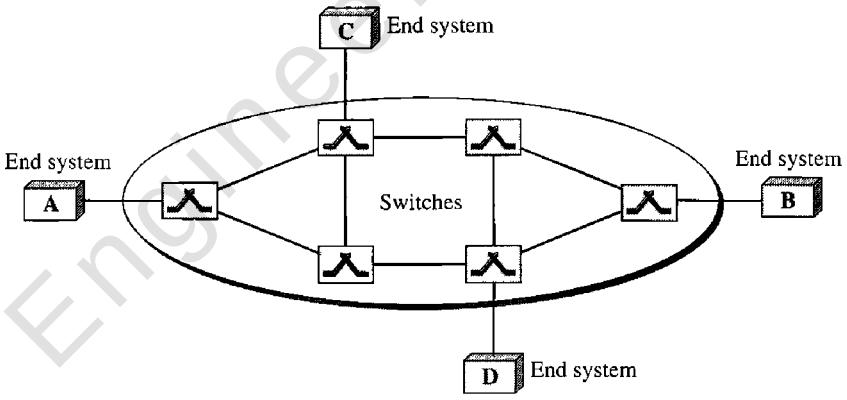
A virtual-circuit network is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.

2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what should be the next switch and the channel on which the packet is being canied), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual-circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure 8.10 is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

Figure 8.10 *Virtual-circuit network*



Addressing

In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

Global Addressing

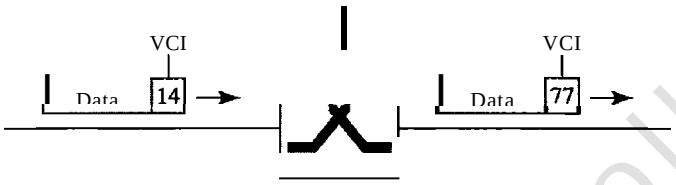
A source or a destination needs to have a global address-an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the virtual-circuit identifier (Vel). A vel, unlike a global address, is a small number that has only switch scope; it

is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCl. Figure 8.11 shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCls.

Figure 8.11 *Virtual-circuit identifier*



Three Phases

As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown. In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection. In the teardown phase, the source and destination inform the switches to delete the corresponding entry. Data transfer occurs between these two phases. We first discuss the data transfer phase, which is more straightforward; we then talk about the setup and teardown phases.

Data Transfer Phase

To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits. Figure 8.12 shows such a switch and its corresponding table.

Figure 8.12 shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

Figure 8.13 shows how a frame from source A reaches destination B and how its VCI changes during the trip. Each switch changes the VCI and routes the frame.

The data transfer phase is active until the source sends all its frames to the destination. The procedure at the switch is the same for each frame of a message. The process creates a virtual circuit, not a real circuit, between the source and destination.

Setup Phase

In the setup phase, a switch creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to B. Two steps are required: the setup request and the acknowledgment.

Figure 8.12 *Switch and tables in a virtual-circuit network*

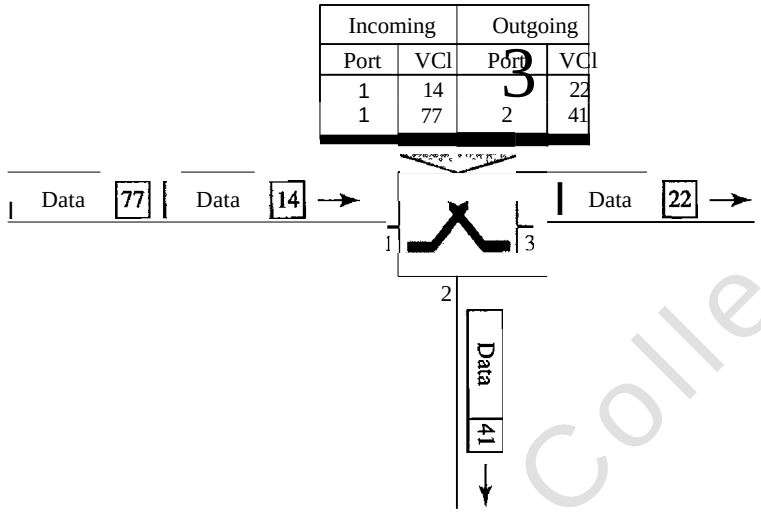
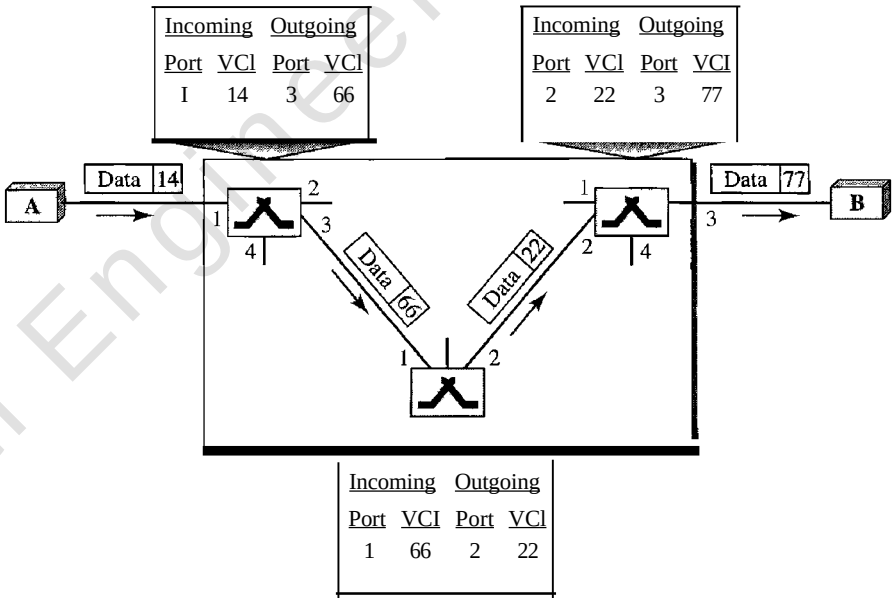


Figure 8.13 *Source-to-destination data transfer in a virtual-circuit network*

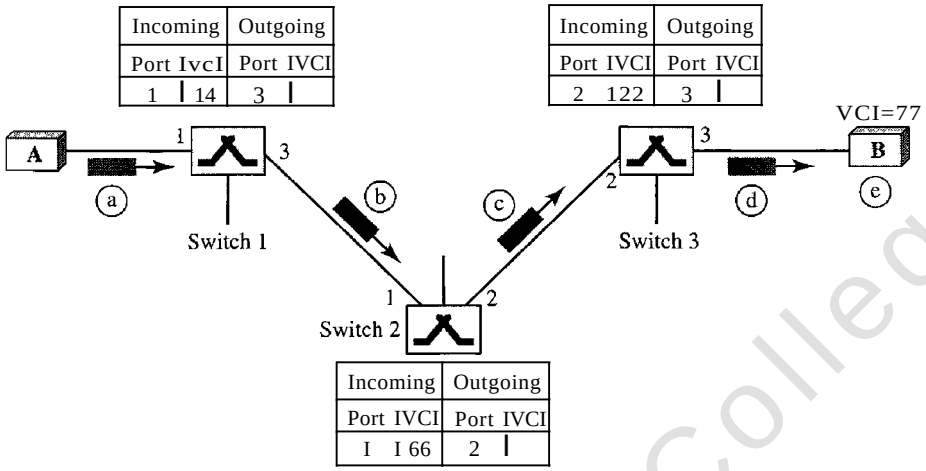


Setup Request A setup request frame is sent from the source to the destination. Figure 8.14 shows the process.

a. Source A sends a setup frame to switch 1.

b. Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; it has a routing table which is different from the switching table. For the moment, assume that it knows the output port. The switch creates an entry in its table for

Figure 8.14 *Setup request in a virtual-circuit network*



this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the outgoing port (3). It does not yet know the outgoing VCI, which will be found during the acknowledgment step. The switch then forwards the frame through port 3 to switch 2.

c. Switch 2 receives the setup request frame. The same events happen here as at switch 1; three columns of the table are completed: in this case, incoming port (l), incoming VCI (66), and outgoing port (2).

d. Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port (3).

e. Destination B receives the setup frame, and if it is ready to receive frames from A, it assigns a VCI to the incoming frames that come from A, in this case 77. This VCI lets the destination know that the frames come from A, and not other sources.

Acknowledgment A special frame, called the acknowledgment frame, completes the entries in the switching tables. Figure 8.15 shows the process.

a. The destination sends an acknowledgment to switch 3. The acknowledgment carries the global source and destination addresses so the switch knows which entry in the table is to be completed. The frame also carries VCI 77, chosen by the destination as the incoming VCI for frames from A. Switch 3 uses this VCI to complete the outgoing VCI column for this entry. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.

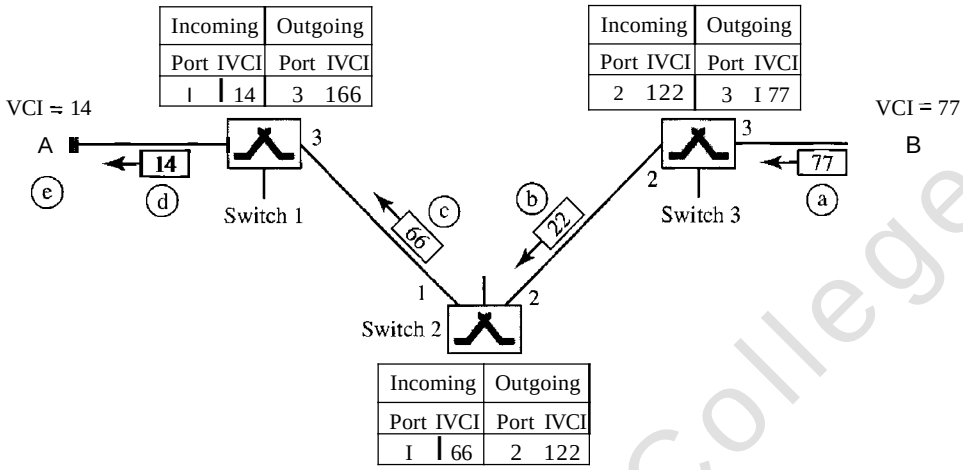
b. Switch 3 sends an acknowledgment to switch 2 that contains its incoming VCI in the table, chosen in the previous step. Switch 2 uses this as the outgoing VCI in the table.

c. Switch 2 sends an acknowledgment to switch 1 that contains its incoming VCI in the table, chosen in the previous step. Switch 1 uses this as the outgoing VCI in the table.

d. Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.

e. The source uses this as the outgoing VCI for the data frames to be sent to destination B.

Figure 8.15 *Setup acknowledgment in a virtual-circuit network*



Teardowil Phase

In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request.* Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

Efficiency

As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays. There is one big advantage in a virtual-circuit network even if resource allocation is on demand. The source can check the availability of the resources, without actually reserving it. Consider a family that wants to dine at a restaurant. Although the restaurant may not accept reservations (allocation of the tables is on demand), the family can call and find out the waiting time. This can save the family time and effort.

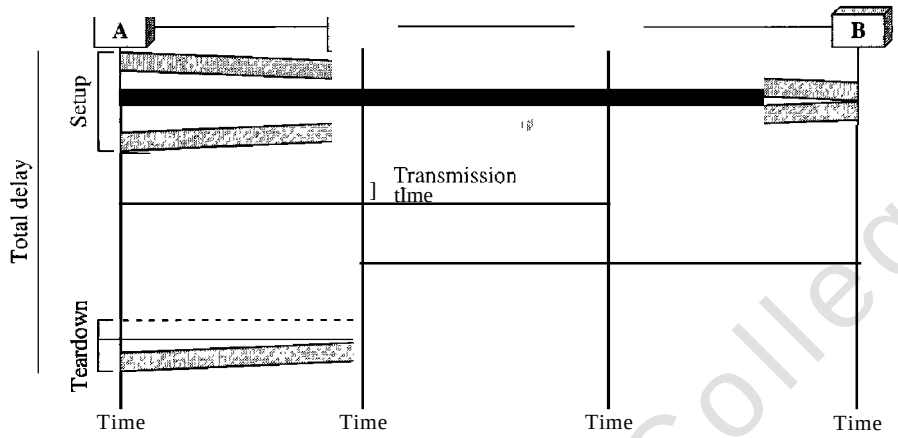
In virtual-circuit switching, all packets belonging to the same source and destination travel the same path; but the packets may arrive at the destination with different delays if resource allocation is on demand.

Delay in Virtual-Circuit Networks

In a virtual-circuit network, there is a one-time delay for setup and a one-time delay for teardown. If resources are allocated during the setup phase, there is no wait time for individual packets. Figure 8.16 shows the delay for a packet traveling through two switches in a virtual-circuit network.

The packet is traveling through two switches (routers). There are three transmission times *(3T),* three propagation times (3τ), data transfer depicted by the sloping lines, a setup delay (which includes transmission and propagation in two directions),

Figure 8.16 *Delay in a virtual-circuit network*



and a teardown delay (which includes transmission and propagation in one direction). We ignore the processing time in each switch. The total delay time is

$$\text{Total delay} = 3T + 3\tau + \text{setup delay} + \text{teardown delay}$$

Circuit-Switched Technology in WANs

As we will see in Chapter 18, virtual-circuit networks are used in switched WANs such as Frame Relay and ATM networks. The data link layer of these technologies is well suited to the virtual-circuit technology.

> Switching at the data link layer in a switched WAN is normally implemented by using virtual-circuit techniques.

8.4 STRUCTURE OF A SWITCH

We use switches in circuit-switched and packet-switched networks. In this section, we discuss the structures of the switches used in each type of network.

Structure of Circuit Switches

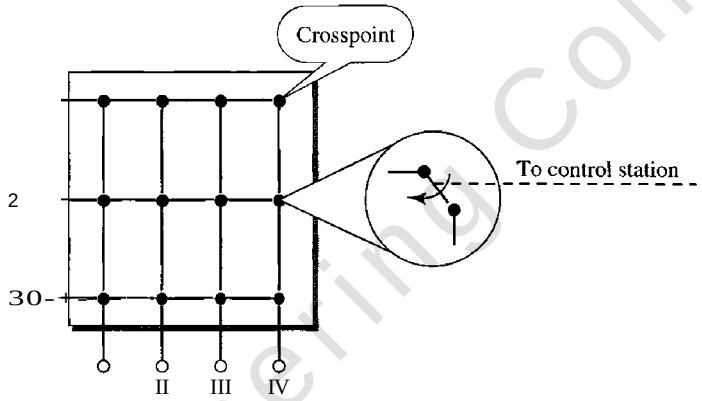
Circuit switching today can use either of two technologies: the space-division switch or the time-division switch.

Space-Division Switch

In space-division switching, the paths in the circuit are separated from one another spatially. This technology was originally designed for use in analog networks but is used currently in both analog and digital networks. It has evolved through a long history of many designs.

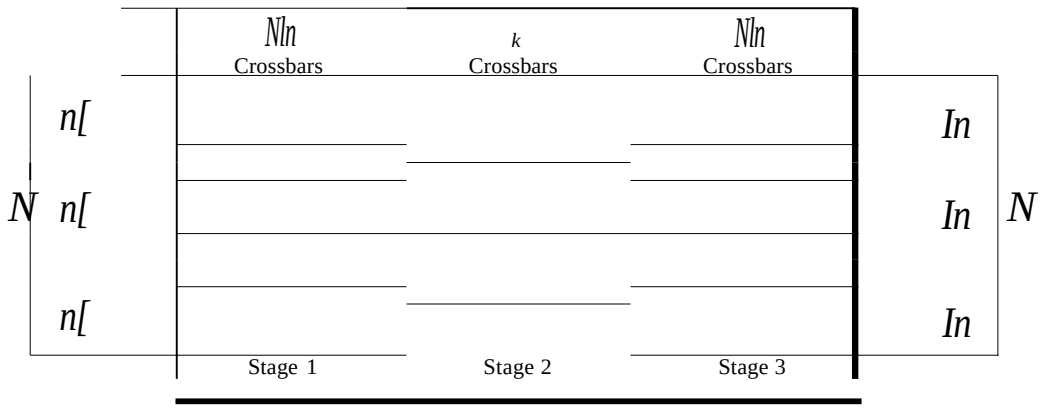
Crossbar Switch A crossbar switch connects *n* inputs to *m* outputs in a grid, using electronic microswitches (transistors) at each crosspoint (see Figure 8.17). The major limitation of this design is the number of crosspoints required. To connect *n* inputs to *m* outputs using a crossbar switch requires *n* x *m* crosspoints. For example, to connect 1000 inputs to 1000 outputs requires a switch with 1,000,000 crosspoints. A crossbar with this number of crosspoints is impractical. Such a switch is also inefficient because statistics show that, in practice, fewer than 25 percent of the crosspoints are in use at any given time. The rest are idle.

Figure 8.17 *Crossbar switch with three inputs and four outputs*



Multistage Switch The solution to the limitations of the crossbar switch is the multistage switch, which combines crossbar switches in several (normally three) stages, as shown in Figure 8.18. In a single crossbar switch, only one row or column (one path) is active for any connection. So we need *N* x N crosspoints. If we can allow multiple paths inside the switch, we can decrease the number of crosspoints. Each crosspoint in the middle stage can be accessed by multiple crosspoints in the first or third stage.

Figure 8.18 *Multistage switch*



To design a three-stage switch, we follow these steps:

1. We divide the N input lines into groups, each of n lines. For each group, we use one crossbar of size n x k, where k is the number of crossbars in the middle stage. In other words, the first stage has N/n crossbars of n x k crosspoints.
2. We use k crossbars, each of size (N/n) x (N/n) in the middle stage.
3. We use N/n crossbars, each of size k x n at the third stage.

We can calculate the total number of crosspoints as follows:

$$\text{Number of crosspoints} = \frac{N}{n}(n \times k) + k\left(\frac{N}{n} \times \frac{N}{n}\right) + \frac{N}{n}(k \times n) = 2kN + k\left(\frac{N}{n}\right)^2$$

In a three-stage switch, the total number of crosspoints is

$$2kN + k\left(\frac{N}{n}\right)^2$$

which is much smaller than the number of crosspoints in a single-stage switch (N^2).

Example 8.3

Design a three-stage, 200 x 200 switch $(N = 200)$ with $k = 4$ and $n = 20$.

Solution

In the first stage we have N/n or 10 crossbars, each of size 20 x 4. In the second stage, we have 4 crossbars, each of size 10 x 10. In the third stage, we have 10 crossbars, each of size 4 x 20. The total number of crosspoints is $2kN + k(N/n)2$, or 2000 crosspoints. This is 5 percent of the number of crosspoints in a single-stage switch (200 x 200 = 40,000).

The multistage switch in Example 8.3 has one drawback-blocking during periods of heavy traffic: The whole idea of multistage switching is to share the crosspoints in the middle-stage crossbars. Sharing can cause a lack of availability if the resources are limited and all users want a connection at the same time. Blocking refers to times when one input cannot be connected to an output because there is no path available between them-all the possible intermediate switches are occupied.

In a single-stage switch, blocking does not occur because every combination of input and output has its own crosspoint; there is always a path. (Cases in which two inputs are trying to contact the same output do not count. That path is not blocked; the output is merely busy.) In the multistage switch described in Example 8.3, however, only 4 of the first 20 inputs can use the switch at a time, only 4 of the second 20 inputs can use the switch at a time, and so on. The small number of crossbars at the middle stage creates blocking.

In large systems, such as those having 10,000 inputs and outputs, the number of stages can be increased to cut down on the number of crosspoints required. As the number of stages increases, however, possible blocking increases as well. Many people have experienced blocking on public telephone systems in the wake of a natural disaster when the calls being made to check on or reassure relatives far outnumber the regular load of the system.

Clos investigated the condition of nonblocking in multistage switches and came up with the following formula. In a nonblocking switch, the number of middle-stage switches must be at least $2n - 1$. In other words, we need to have $k \geq 2n - 1$.

Note that the number of crosspoints is still smaller than that in a single-stage switch. Now we need to minimize the number of crosspoints with a fixed N by using the Clos criteria. We can take the derivative of the equation with respect to n (the only variable) and find the value of n that makes the result zero. This n must be equal to or greater than $(N/2)1/2$. In this case, the total number of crosspoints is greater than or equal to $4N [(2N) 112 - 1]$. In other words, the minimum number of crosspoints according to the Clos criteria is proportional to $N^3/2$.

According to Clos criterion:

$n = (NI2)1/2$

$k > 2n-1$

Total number of crosspoints $\geq 4N$ [(2N)1/2 - 1]

Example 8.4

Redesign the previous three-stage, 200 x 200 switch, using the Clos criteria with a minimum number of crosspoints.

Solution

We let $n = (200/2)1/2$, or $n = 10$. We calculate $k = 2n - 1 = 19$. In the first stage, we have 200/10, or 20, crossbars, each with lOX 19 crosspoints. In the second stage, we have 19 crossbars, each with 10 x 10 crosspoints. In the third stage, we have 20 crossbars each with 19 x 10 crosspoints. The total number of crosspoints is 20(10 X 19) + 19(10 X 10) + 20(19 XlO) = 9500. If we use a single-stage switch, we need 200 x 200 = 40,000 crosspoints. The number of crosspoints in this three-stage switch is 24 percent that of a single-stage switch. More points are needed than in Example 8.3 (5 percent). The extra crosspoints are needed to prevent blocking.

A multistage switch that uses the Clos criteria and a minimum number of crosspoints still requires a huge number of crosspoints. For example, to have a 100,000 input/output switch, we need something close to 200 million crosspoints (instead of 10 billion). This means that if a telephone company needs to provide a switch to connect 100,000 telephones in a city, it needs 200 million crosspoints. The number can be reduced if we accept blocking. Today, telephone companies use time-division switching or a combination of space- and time-division switches, as we will see shortly.

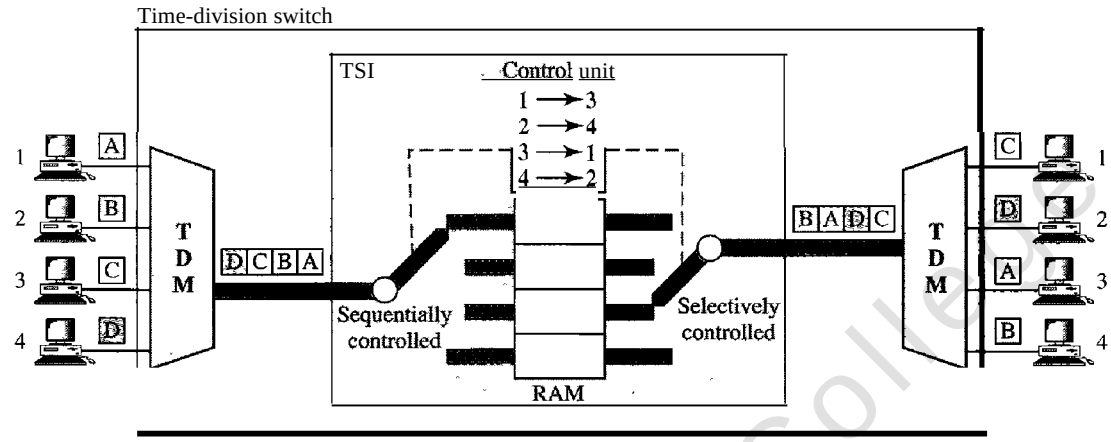
Time-Division Switch

Time-division switching uses time-division multiplexing (TDM) inside a switch. The most popular technology is called the time-slot interchange (TSI).

Time-Slot Interchange Figure 8.19 shows a system connecting four input lines to four output lines. Imagine that each input line wants to send data to an output line according to the following pattern:

$1 \rightarrow 3 \quad 2 \rightarrow 4 \quad 3 \rightarrow 1 \quad 4 \rightarrow 2$

Figure 8.19 *Time-slot interchange*



The figure combines a TDM multiplexer, a TDM demultiplexer, and a TSI consisting of random access memory (RAM) with several memory locations. The size of each location is the same as the size of a single time slot. The number oflocations is the same as the number of inputs (in most cases, the numbers of inputs and outputs are equal). The RAM fills up with incoming data from time slots in the order received. Slots are then sent out in an order based on the decisions of a control unit.

Time- and Space-Division Switch Combinations

When we compare space-division and time-division switching, some interesting facts emerge. The advantage of space-division switching is that it is instantaneous. Its disadvantage is the number of crosspoints required to make space-division switching acceptable in terms of blocking.

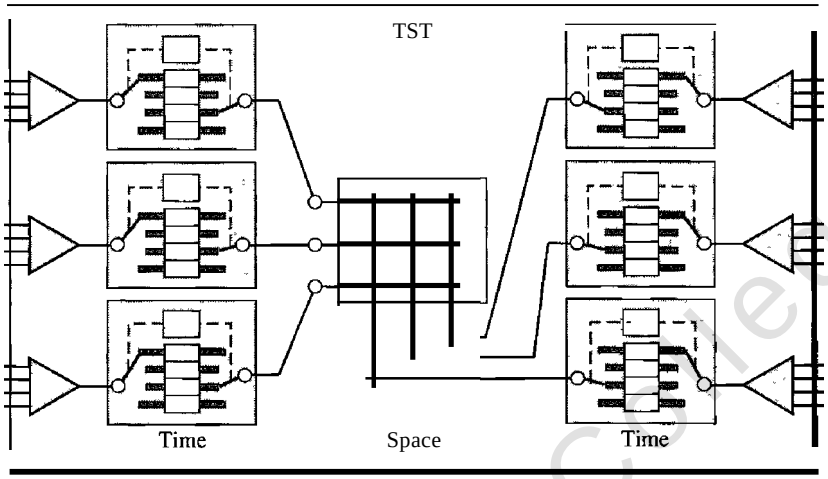
The advantage of time-division switching is that it needs no crosspoints. Its disadvantage, in the case of TSI, is that processing each connection creates delays. Each time slot must be stored by the RAM, then retrieved and passed on.

In a third option, we combine space-division and time-division technologies to take advantage of the best of both. Combining the two results in switches that are optimized both physically (the number of crosspoints) and temporally (the amount of delay). Multistage switches of this sort can be designed as time-space-time (TST) switch.

Figure 8.20 shows a simple TST switch that consists of two time stages and one space stage and has 12 inputs and 12 outputs. Instead of one time-division switch, it divides the inputs into three groups (of four inputs each) and directs them to three time-slot interchanges. The result is that the average delay is one-third of what would result from using one time-slot interchange to handle all 12 inputs.

The last stage is a mirror image of the first stage. The middle stage is a space-division switch (crossbar) that connects the TSI groups to allow connectivity between all possible input and output pairs (e.g., to connect input 3 of the first group to output 7 of the second group).

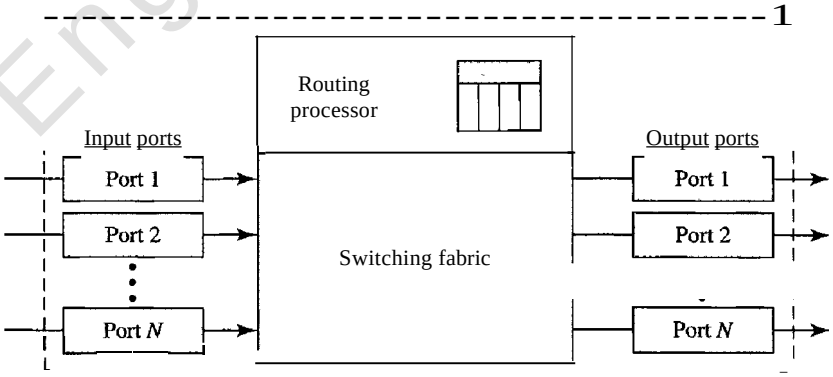
Figure 8.20 *Time-space-time switch*



Structure of Packet Switches

A switch used in a packet-switched network has a different structure from a switch used in a circuit-switched network.We can say that a packet switch has four components: input ports, output ports, the routing processor, and the switching fabric, as shown in Figure 8.21.

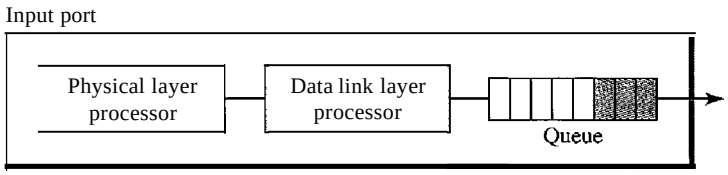
Figure 8.21 *Packet switch components*



Input Ports

An input port performs the physical and data link functions of the packet switch. The bits are constructed from the received signal. The packet is decapsulated from the frame. Errors are detected and corrected. The packet is now ready to be routed by the network layer. In addition to a physical layer processor and a data link processor, the input port has buffers (queues) to hold the packet before it is directed to the switching fabric. Figure 8.22 shows a schematic diagram of an input port.

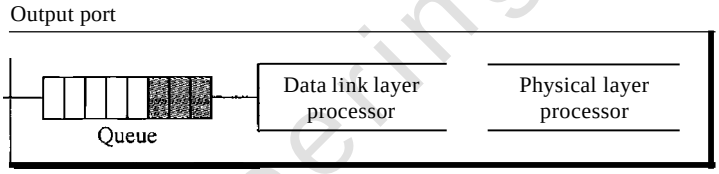
Figure 8.22 *Input port*



Output Port

The output port performs the same functions as the input port, but in the reverse order. First the outgoing packets are queued, then the packet is encapsulated in a frame, and finally the physical layer functions are applied to the frame to create the signal to be sent on the line. Figure 8.23 shows a schematic diagram of an output port.

Figure 8.23 *Output port*



ROllting Processor

The routing processor performs the functions of the network layer. The destination address is used to find the address of the next hop and, at the same time, the output port number from which the packet is sent out. This activity is sometimes referred to as **table lookup** because the routing processor searches the routing table. In the newer packet switches, this function of the routing processor is being moved to the input ports to facilitate and expedite the process.

Switching Fabrics

The most difficult task in a packet switch is to move the packet from the input queue to the output queue. The speed with which this is done affects the size of the input/output queue and the overall delay in packet delivery. In the past, when a packet switch was actually a dedicated computer, the memory of the computer or a bus was used as the switching fabric. The input port stored the packet in memory; the output port retrieved the packet from memory. Today, packet switches are specialized mechanisms that use a variety of switching fabrics. We briefly discuss some of these fabrics here.

Crossbar Switch The simplest type of switching fabric is the crossbar switch, discussed in the previous section.

Banyan Switch A more realistic approach than the crossbar switch is the banyan switch (named after the banyan tree). A banyan switch is a multistage switch with

microswitches at each stage that route the packets based on the output port represented as a binary string. For *n* inputs and *n* outputs, we have log2 *n* stages with *nl2* microswitches at each stage. The first stage routes the packet based on the high-order bit of the binary string. The second stage routes the packet based on the second high-order bit, and so on. Figure 8.24 shows a banyan switch with eight inputs and eight outputs. The number of stages is $\log_2(8) = 3$.

Figure 8.24 *A banyan switch*

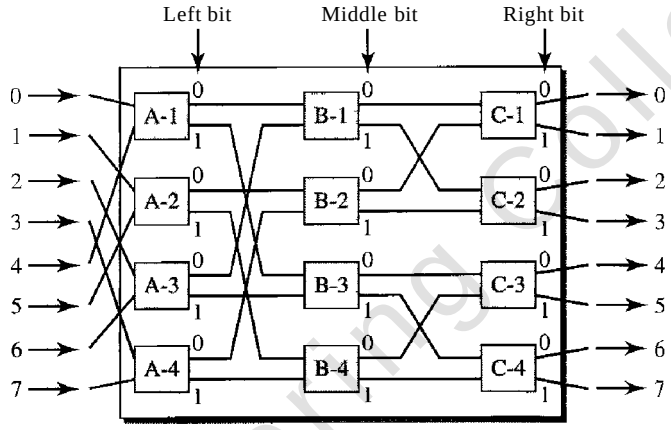
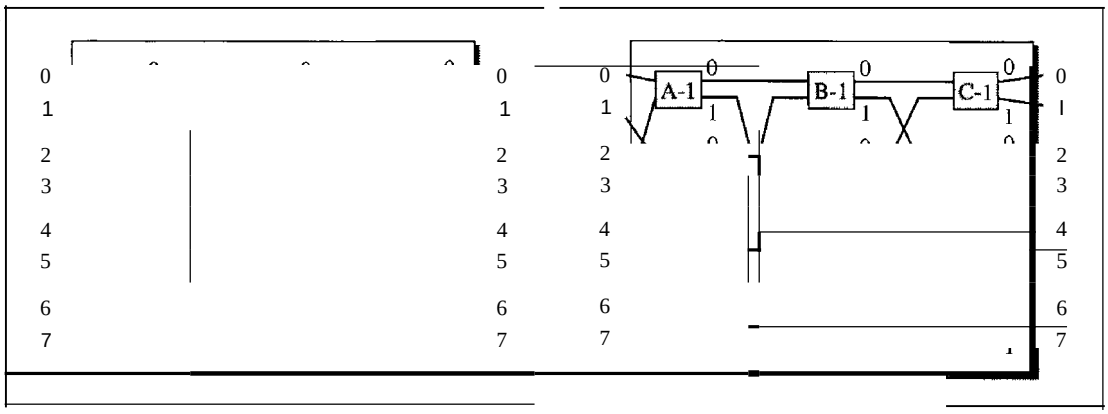


Figure 8.25 shows the operation. In part a, a packet has arrived at input port 1 and must go to output port 6 (110 in binary). The first microswitch (A-2) routes the packet based on the first bit (1), the second microswitch (B-4) routes the packet based on the second bit (1), and the third microswitch (C-4) routes the packet based on the third bit (0). In part b, a packet has arrived at input port 5 and must go to output port 2 (010 in binary). The first microswitch (A-2) routes the packet based on the first bit (0), the second microswitch (B-2) routes the packet based on the second bit (1), and the third microswitch (C-2) routes the packet based on the third bit (0).

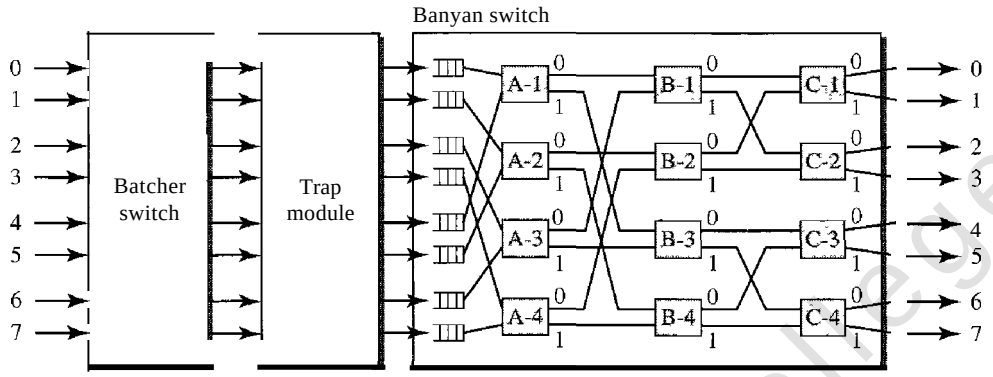
Figure 8.25 *Examples of routing in a banyan switch*



a. Input 1 sending a cell to output 6 (110)

b. Input 5 sending a cell to output 2 (010)

Figure 8.26 *Batcher-banyan switch*



Batcher-Banyan Switch The problem with the banyan switch is the possibility of internal collision even when two packets are not heading for the same output port. We can solve this problem by sorting the arriving packets based on their destination port.

K. E. Batcher designed a switch that comes before the banyan switch and sorts the incoming packets according to their final destinations. The combination is called the **Batcher-banyan switch.** The sorting switch uses hardware merging techniques, but we do not discuss the details here. Normally, another hardware module called a **trap** is added between the Batcher switch and the banyan switch (see Figure 8.26) The trap module prevents duplicate packets (packets with the same output destination) from passing to the banyan switch simultaneously. Only one packet for each destination is allowed at each tick; if there is more than one, they wait for the next tick.

8.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Switching is discussed in Chapter 10 of [Sta04] and Chapters 4 and 7 of [GW04]. Circuit-switching is fully discussed in [BELOO].

8.6 KEY TERMS

banyan switch
Batcher-banyan switch
blocking
circuit switching
circuit-switched network
crossbar switch
crosspoint
data transfer phase
datagram
datagram network
end system
input port

multistage switch
output port
packet-switched network
routing processor
setup phase
space-division switching
switch
switching
switching fabric
table lookup
teardown phase
time-division switching
time-slot interchange (TSI)
time-space-time (TST) switch
trap
virtual-circuit identifier (VCI)
virtual-circuit network

8.7 SUMMARY

- A switched network consists of a series of interlinked nodes, called switches. Traditionally' three methods of switching have been important: circuit switching, packet switching, and message switching.
- We can divide today's networks into three broad categories: circuit-switched networks, packet-switched networks, and message-switched. Packet-switched networks can also be divided into two subcategories: virtual-circuit networks and datagram networks
- A circuit-switched network is made of a set of switches connected by physical links, in which each link is divided into n channels. Circuit switching takes place at the physical layer. In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer phase until the teardown phase.
- In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand.
- In a datagram network, each packet is treated independently of all others. Packets in this approach are referred to as datagrams. There are no setup or teardown phases.
- A virtual-circuit network is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.
- Circuit switching uses either of two technologies: the space-division switch or the time-division switch.
- A switch in a packet-switched network has a different structure from a switch used in a circuit-switched network.We can say that a packet switch has four types of components: input ports, output ports, a routing processor, and switching fabric.

8.8 PRACTICE SET

Review Questions

1. Describe the need for switching and define a switch.
2. List the three traditional switching methods. What are the most common today?

3. What are the two approaches to packet-switching?
4. Compare and contrast a circuit-switched network and a packet-switched network.
5. What is the role of the address field in a packet traveling through a datagram network?
6. What is the role of the address field in a packet traveling through a virtual-circuit network?
7. Compare space-division and time-division switches.
8. What is TSI and its role in a time-division switching?
9. Define blocking in a switched network.
10. List four major components of a packet switch and their functions.

Exercises

11. A path in a digital circuit-switched network has a data rate of l Mbps. The exchange of 1000 bits is required for the setup and teardown phases. The distance between two parties is 5000 km. Answer the following questions if the propagataion speed is 2×10^8 m:
    a. What is the total delay if 1000 bits of data are exchanged during the data transfer phase?
    b. What is the total delay if 100,000 bits of data are exchanged during the data transfer phase?
    c. What is the total delay if 1,000,000 bits of data are exchanged during the data transfer phase?
    d. Find the delay per 1000 bits of data for each of the above cases and compare them. What can you infer?
12. Five equal-size datagrams belonging to the same message leave for the destination one after another. However, they travel through different paths as shown in Table 8.1.

Table 8.1 *Exercise 12*

Datagram	*Path Length*	*Visited Switches*
1	3200Km	1,3,5
2	11,700 Km	1,2,5
3	12,200 Km	1,2,3,5
4	10,200 Km	1,4,5
5	10,700 Km	1,4,3,5

We assume that the delay for each switch (including waiting and processing) is 3, 10, 20, 7, and 20 ms respectively. Assuming that the propagation speed is 2×10^8 m, find the order the datagrams arrive at the destination and the delay for each. Ignore any other delays in transmission.

13. Transmission of information in any network involves end-to-end addressing and sometimes local addressing (such as YCI). Table 8.2 shows the types of networks and the addressing mechanism used in each of them.

Table 8.2 *Exercise 13*

Network	*Setup*	*Data Transfer*	*Teardown*
Circuit-switched	End-ta-end		End-ta-end
Datagram		End-ta-end	
Virtual-circuit	End-to-end	Local	End-to-end

Answer the following questions:

a. Why does a circuit-switched network need end-to-end addressing during the setup and teardown phases? Why are no addresses needed during the data transfer phase for this type of network?

b. Why does a datagram network need only end-to-end addressing during the data transfer phase, but no addressing during the setup and teardown phases?

c. Why does a virtual-circuit network need addresses during all three phases?

14. We mentioned that two types of networks, datagram and virtual-circuit, need a routing or switching table to find the output port from which the information belonging to a destination should be sent out, but a circuit-switched network has no need for such a table. Give the reason for this difference.

15. An entry in the switching table of a virtual-circuit network is normally created during the setup phase and deleted during the teardown phase. In other words, the entries in this type of network reflect the current connections, the activity in the network. In contrast, the entries in a routing table of a datagram network do not depend on the current connections; they show the configuration of the network and how any packet should be routed to a final destination. The entries may remain the same even if there is no activity in the network. The routing tables, however, are updated if there are changes in the network. Can you explain the reason for these two different characteristics? Can we say that a virtual-circuit is a *connection-oriented* network and a datagram network is a *connectionLess* network because of the above characteristics?

16. The minimum number of columns in a datagram network is two; the minimum number of columns in a virtual-circuit network is four. Can you explain the reason? Is the difference related to the type of addresses carried in the packets of each network?

17. Figure 8.27 shows a switch (router) in a datagram network.

    Find the output port for packets with the following destination addresses:
    Packet 1: 7176
    Packet 2: 1233
    Packet 3: 8766
    Packet 4: 9144

18. Figure 8.28 shows a switch in a virtual circuit network.

Figure 8.27 *Exercise 17*

Destination address	Output port
1233	3
1456	2
3255	1
4470	4
7176	2
8766	3
9144	2

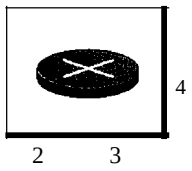
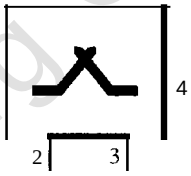


Figure 8.28 *Exercise 18*

Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
2	71	4	41
2	92	1	45
3	58	2	43
3	78	2	70
4	56	3	11



Find the output port and the output VCI for packets with the following input port and input VCI addresses:
Packet 1: 3, 78
Packet 2: 2, 92
Packet 3: 4, 56
Packet 4: 2, 71

19. Answer the following questions:
    a. Can a routing table in a datagram network have two entries with the same destination address? Explain.
    b. Can a switching table in a virtual-circuit network have two entries with the same input port number? With the same output port number? With the same incoming VCls? With the same outgoing VCls? With the same incoming values (port, VCI)? With the same outgoing values (port, VCI)?
20. It is obvious that a router or a switch needs to do searching to find information in the corresponding table. The searching in a routing table for a datagram network is based on the destination address; the searching in a switching table in a virtual-circuit network is based on the combination of incoming port and incoming VCI. Explain the reason and define how these tables must be ordered (sorted) based on these values.
21. Consider an n x k crossbar switch with n inputs and k outputs.
    a. Can we say that switch acts as a multiplexer if $n > k$?
    b. Can we say that switch acts as a demultiplexer if $n < k$?

22. We need a three-stage space-division switch with N = 100. We use 10 crossbars at the first and third stages and 4 crossbars at the middle stage.
    a. Draw the configuration diagram.
    b. Calculate the total number of crosspoints.
    c. Find the possible number of simultaneous connections.
    d. Find the possible number of simultaneous connections if we use one single crossbar (100 x 100).
    e. Find the blocking factor, the ratio of the number of connections in c and in d.
23. Repeat Exercise 22 if we use 6 crossbars at the middle stage.
24. Redesign the configuration of Exercise 22 using the Clos criteria.
25. We need to have a space-division switch with 1000 inputs and outputs. What is the total number of crosspoints in each of the following cases?
    a. Using one single crossbar.
    b. Using a multi-stage switch based on the Clos criteria
26. We need a three-stage time-space-time switch with N = 100. We use 10 TSIs at the first and third stages and 4 crossbars at the middle stage.
    a. Draw the configuration diagram.
    b. Calculate the total number of crosspoints.
    c. Calculate the total number of memory locations we need for the TSIs.

PART 3

Data Link Layer

Objectives

The data link layer transforms the physical layer, a raw transmission facility, to a link responsible for node-to-node (hop-to-hop) communication. Specific responsibilities of the data link layer include *framing, addressing, flow control, error control,* and *media access control.* The data link layer divides the stream of bits received from the network layer into manageable data units called frames. The data link layer adds a header to the frame to define the addresses of the sender and receiver of the frame. If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver. The data link layer also adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged, duplicate, or lost frames. When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

In Part 3 of the book, we first discuss services provided by the data link layer. We then discuss the implementation of these services in local area networks (LANs). Finally we discuss how wide area networks (WANs) use these services.

Part 3 of the book is devoted to the data link layer and the services provided by this layer.

Chapters

This part consists of nine chapters: Chapters 10 to 18.

Chapter 10

Chapter 10 discusses error detection and correction. Although the quality of devices and media have been improved during the last decade, we still need to check for errors and correct them in most applications.

Chapter 11

Chapter 11 is named data link control, which involves flow and error control. It discusses some protocols that are designed to handle the services required from the data link layer in relation to the network layer.

Chapter 12

Chapter 12 is devoted to access control, the duties of the data link layer that are related to the use of the physical layer.

Chapter 13

This chapter introduces wired local area networks. A wired LAN, viewed as a link, is mostly involved in the physical and data link layers. We have devoted the chapter to the discussion of Ethernet and its evolution, a dominant technology today.

Chapter 14

This chapter introduces wireless local area networks. The wireless LAN is a growing technology in the Internet. We devote one chapter to this topic.

Chapter 15

After discussing wired and wireless LANs, we show how they can be connected together using connecting devices.

Chapter 16

This is the first chapter on wide area networks (WANs). We start with wireless WANs and then move on to satellite networks and mobile telephone networks.

Chapter 17

To demonstrate the operation of a high-speed wide area network that can be used as a backbone for other WANs or for the Internet, we have chosen to devote all of Chapter 17 to SONET, a wide area network that uses fiber-optic technology.

Chapter 18

This chapter concludes our discussion on wide area networks. Two switched WANs, Frame Relay and ATM, are discussed here.

CHAPTER 10

Error Detection and Correction

Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting errors.

Data can be corrupted during transmission.
Some applications require that errors be detected and corrected.

Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy.

10.1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correcion.

Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a 0. In a burst error, multiple bits are changed. For example, a 11100 s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the 12 bits of information.

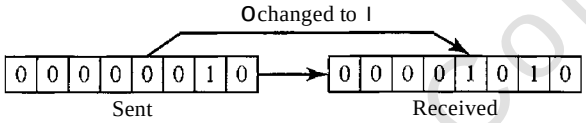
Single-Bit Error

The term *single-bit error* means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

In a single-bit error, only 1 bit in the data unit has changed.

Figure 10.1 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In Figure 10.1,00000010 (ASCII *STX*) was sent, meaning *start of text,* but 00001010 (ASCII *LF)* was received, meaning *line feed.* (For more information about ASCII code, see Appendix A.)

:Figure 10.1 *Single-bit error*



Single-bit errors are the least likely type of error in serial data transmission. To understand why, imagine data sent at 1 Mbps. This means that each bit lasts only 1/1,000,000 s, or 1 μs. For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare; noise normally lasts much longer than this.

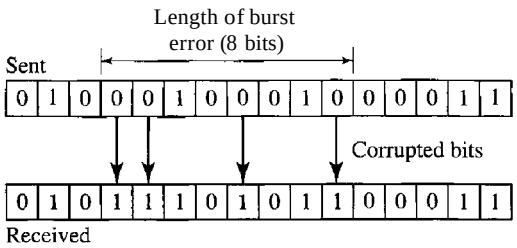
Burst Error

The term *burst error* means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

A burst error means that 2 or more bits in the data unit have changed.

Figure 10.2 shows the effect of a burst error on a data unit. In this case, 0100010001000011 was sent, but 0101110101100011 was received. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

Figure 10.2 *Burst error of length 8*



A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at I kbps, a noise of 11100 s can affect 10 bits; if we are sending data at I Mbps, the same noise can affect 10,000 bits.

Redundancy

The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

To detect or correct errors, we need to send **extra** (redundant) bits with data.

Detection Versus Correction

The correction of errors is more difficult than the detection. In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Forward Error Correction Versus Retransmission

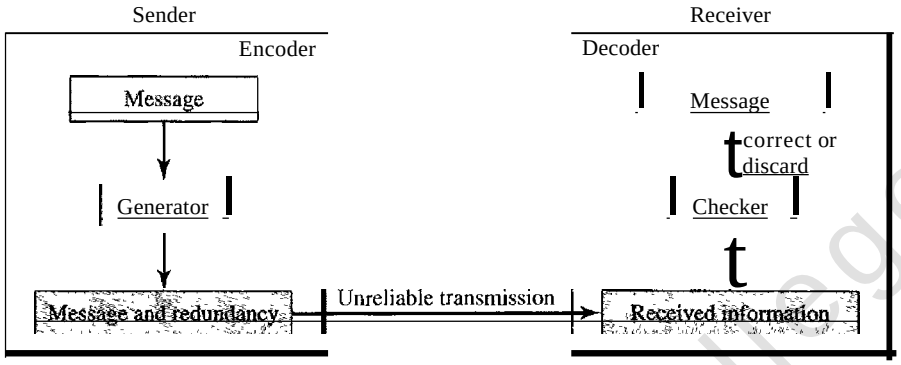
There are two main methods of error correction. Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small. Correction by retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free (usually, not all errors can be detected).

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors. The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme. Figure 10.3 shows the general idea of coding.

We can divide coding schemes into two broad categories: block coding and convolution coding. In this book, we concentrate on block coding; convolution coding is more complex and beyond the scope of this book.

Figure 10.3 *The structure of encoder and decoder*



In this book, we concentrate on block codes; we leave convolution codes to advanced texts.

Modular Arithmetic

Before we finish this section, let us briefly discuss a concept basic to computer science in general and to error detection and correction in particular: modular arithmetic. Our intent here is not to delve deeply into the mathematics of this topic; we present just enough information to provide a background to materials discussed in this chapter.

In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus *N.* We then use only the integers 0 to *N* - I, inclusive. This is *modulo-N* arithmetic. For example, if the modulus is 12, we use only the integers 0 to 11, inclusive. An example of modulo arithmetic is our clock system. It is based on modulo-12 arithmetic, substituting the number 12 for 0. In a *modulo-N* system, if a number is greater than *N,* it is divided by *N* and the remainder is the result. If it is negative, as many *Ns* as needed are added to make it positive. Consider our clock system again. If we start a job at 11 A.M. and the job takes 5 h, we can say that the job is to be finished at 16:00 if we are in the military, or we can say that it will be finished at 4 P.M. (the remainder of 16/12 is 4).

In modulo-N arithmetic, we use only the integers in the range 0 to *N* — 1, inclusive.

Addition and subtraction in modulo arithmetic are simple. There is no carry when you add two digits in a column. There is no carry when you subtract one digit from another in a column.

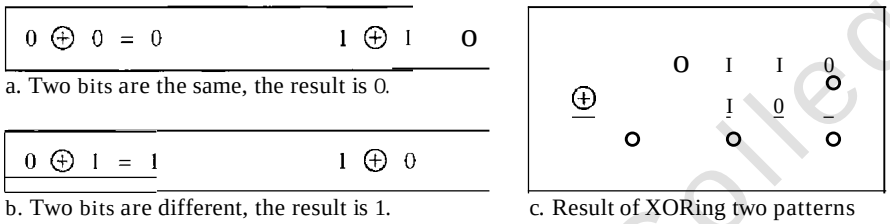
Modulo-2 Arithmetic

Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus *N* is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.

Adding:	0+0=0	0+1=1	1+0=1	1+1=0
Subtracting:	0-0=0	0-1=1	1 – 0 = 1	1-1=0

Notice particularly that addition and subtraction give the same results. In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is I if two bits are different. Figure 10.4 shows this operation.

Figure **10.4** *XORing of two single bits or two words*



a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.

c. Result of XORing two patterns

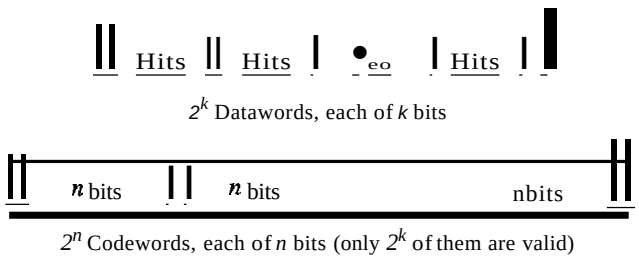
Other Modulo Arithmetic

We also use, modulo-N arithmetic through the book. The principle is the same; we use numbers between 0 and $N - 1$. If the modulus is not 2, addition and subtraction are distinct. If we get a negative result, we add enough multiples of N to make it positive.

10.2 BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called codewords. How the extra r bits is chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of datawords, each of size k, and a set of codewords, each of size of n. With k bits, we can create a combination of 2^k datawords; with n bits, we can create a combination of 2^n codewords. Since $n > k$, the number of possible codewords is larger than the number of possible datawords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2^n - 2^k$ codewords that are not used. We call these codewords invalid or illegal. Figure 10.5 shows the situation.

Figure 10.5 *Datawords and codewords in block coding*



Example 10.1

The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, $k = 4$ and $n = 5$. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

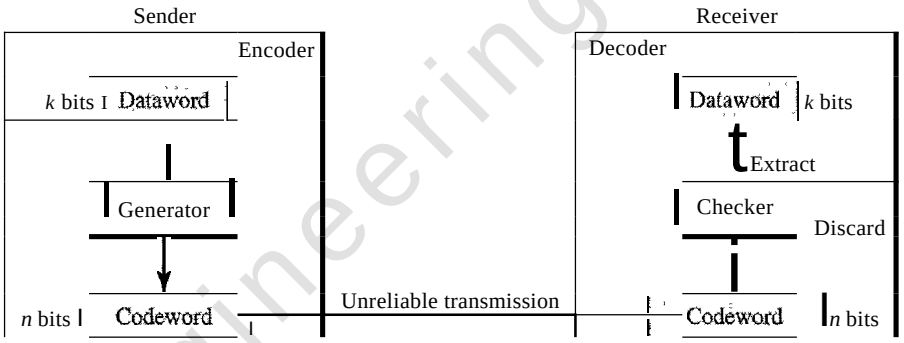
Error Detection

How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.
2. The original codeword has changed to an invalid one.

Figure 10.6 shows the role of block coding in error detection.

Figure 10.6 *Process of error detection in block coding*



The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding (discussed later). Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected. This type of coding can detect only single errors. Two or more errors may remain undetected.

Example 10.2

Let us assume that $k = 2$ and $n = 3$. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Table **10.1** *A code for error detection (Example 10.2)*

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

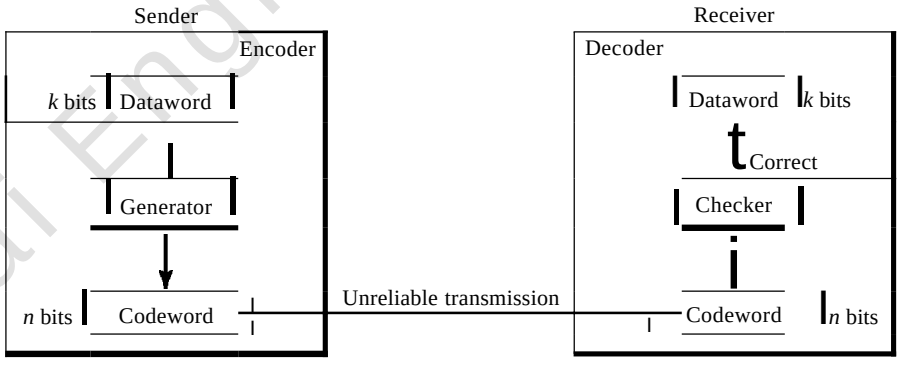
1. The receiver receives OIl. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Error Correction

As we said before, error correction is much more difficult than error detection. In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent. We can say that we need more redundant bits for error correction than for error detection. Figure 10.7 shows the role of block coding in error correction. We can see that the idea is the same as error detection but the checker functions are much more complex.

Figure 10.7 *Structure of encoder and decoder in error correction*



Example 10.3

Let us add more redundant bits to Example 10.2 to see if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords. Again, later we will show how we chose the redundant bits. For the moment let us concentrate on the error correction concept. Table 10.2 shows the datawords and codewords.

Assume the dataword is 01. The sender consults the table (or uses an algorithm) to create the codeword 01011. The codeword is corrupted during transmission, and 01001 is received (error in the second bit from the right). First, the receiver finds that the received codeword is not in the table. This means an error has occurred. (Detection must come before correction.) The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

Table 10.2 *A code for error correction (Example 10.3)*

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

I. Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.
2. By the same reasoning, the original codeword cannot be the third or fourth one in the table.
3. The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.

Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as $d(x, y)$.

The Hamming distance can easily be found if wc apply the XOR operation ($\oplus$) on the two words and count the number of Is in the result. Note that the Hamming distance is a value greater than zero.

The Hamming distance between two words is the number of differences between corresponding bits.

Example 10.4

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance $d(OOO, 011)$ is 2 because $000 \oplus 011$ is 011 (two Is).
2. The Hamming distance $d(10101, 11110)$ is 3 because $10101 \oplus 11110$ is 01011 (three Is).

Minimum Hamming Distance

Although the concept of the Hamming distance is the central point in dealing with error detection and correction codes, the measurement that is used for designing a code is the minimum Hamming distance. In a set of words, the minimum Hamming distance is the smallest Hamming distance between all possible pairs. We use d_{min} to define the minimum Hamming distance in a coding scheme. To find this value, we find the Hamming distances between all words and select the smallest one.

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example 10.5

Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

$$d(000, 011) = 2 \quad d(000, 101) = 2 \quad d(0a0, 110) = 2 \quad d(011, 101) = 2$$
$$d(011, 110) = 2 \quad d(w1, 110) = 2$$

The d_{min} in this case is 2.

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

$$d(00000, 01011) = 3 \quad d(00000, 10101) = 3 \quad d(00000, 11110) = 4$$
$$d(01011, 10101) = 4 \quad d(0l011, 11110) = 3 \quad d(10101, 11110) = 3$$

The d_{min} in this case is 3.

Three Parameters

Before we continue with our discussion, we need to mention that any coding scheme needs to have at least three parameters: the codeword size *n*, the dataword size *k*, and the minimum Hamming distance d_{min}. A coding scheme C is written as *C(n, k)* with a separate expression for d_{min}. For example, we can call our first coding scheme *C(3, 2)* with $d_{min} = 2$ and our second coding scheme *C(5, 2)* with $d_{min} = 3$.

Hamming Distance and Error

Before we explore the criteria for error detection or correction, let us discuss the relationship between the Hamming distance and errors occurring during transmission. When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error. In other words, the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $d(00000, 01101) = 3$.

Minimum Distance for Error Detection

Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to detect up to s errors, the minimum distance between the valid codes must be s + 1, so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is s + 1, the received codeword cannot be erroneously mistaken for another codeword. The distances are not enough (s + 1) for the receiver to accept it as valid. The error will be detected. We need to clarify a point here: Although a code with $d_{min} = s + 1$

may be able to detect more than s errors in some special cases, only s or fewer errors are guaranteed to be detected.

To guarantee the detection of up to *s* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Example 10.7

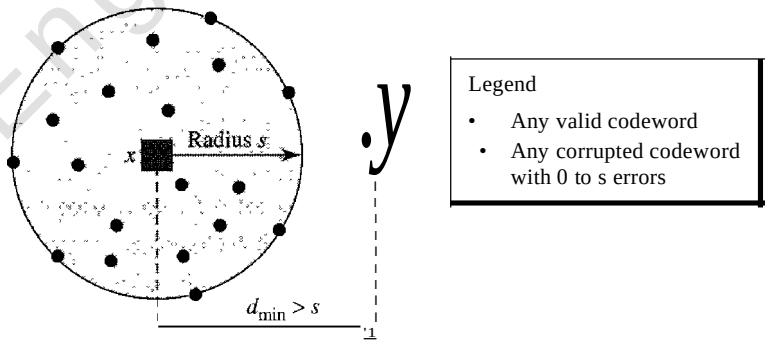
The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Example 10.8

Our second block code scheme (Table 10.2) has $d_{min} = 3$. This code can detect up to two errors. Again, we see that when any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled. However, some combinations of three errors change a valid codeword to another valid codeword. The receiver accepts the received codeword and the errors are undetected.

We can look at this geometrically. Let us assume that the sent codeword *x* is at the center of a circle with radius *s*. All other received codewords that are created by 1 to *s* errors are points inside the circle or on the perimeter of the circle. All other valid codewords must be outside the circle, as shown in Figure 10.8.

Figure 10.8 *Geometric concept for finding d_{min} in error detection*



In Figure 10.8, d_{min} must be an integer greater than *s*; that is, $d_{min} = s + 1$.

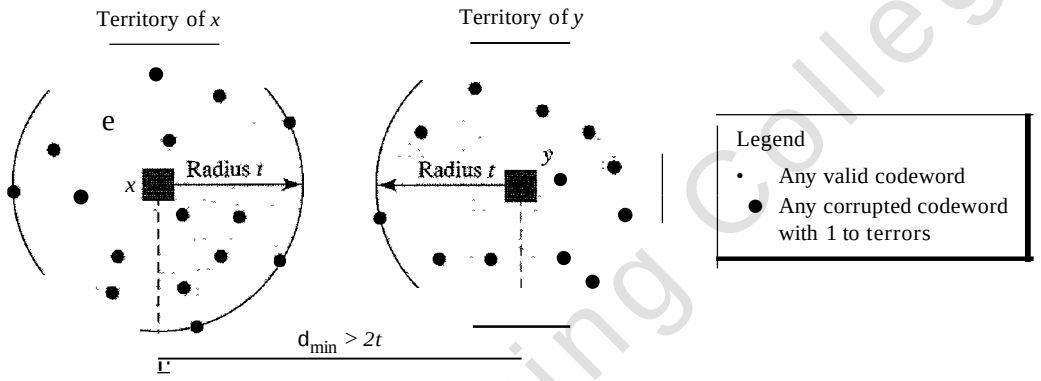
Minimum Distance for Error Correction

Error correction is more complex than error detection; a decision is involved. When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword. Each valid codeword has its own territory.

We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius of *t* and that the valid codeword is at the

center. For example, suppose a codeword x is corrupted by t bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the center. Note that we assume that only up to t errors have occurred; otherwise, the decision is wrong. Figure 10.9 shows this geometric interpretation. Some texts use a sphere to show the distance between all valid block codes.

Figure 10.9 *Geometric concept for finding d_{min} in error correction*



In Figure 10.9, $d_{min} > 2t$; since the next integer increment is 1, we can say that $d_{min} = 2t + 1$.

To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

Example 10.9

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes need to have an odd minimum distance (3, 5, 7, ...).

10.3 LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes. The use of nonlinear block codes for error detection and correction is not as widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes.

The formal definition of linear block codes requires the knowledge of abstract algebra (particularly Galois fields), which is beyond the scope of this book. We therefore give an informal definition. For our purposes, a linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

> In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Example 10.10

Let us see if the two codes we defined in Table 10.1 and Table 10.2 belong to the class of linear block codes.

1. The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.
2. The scheme in Table 10.2 is also a linear block code. We can create all four codewords by XORing two other codewords.

Minimum Distance for Linear Block Codes

It is simple to find the minimum Hamming distance for a linear block code. The minimum Hamming distance is the number of Is in the nonzero valid codeword with the smallest number of Is.

Example 10.11

In our first code (Table 10.1), the numbers of Is in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$. In our second code (Table 10.2), the numbers of Is in the nonzero codewords are 3, 3, and 4. So in this code we have $d_{min} = 3$.

Some Linear Block Codes

Let us now show some linear block codes. These codes are trivial because we can easily find the encoding and decoding algorithms and check their performances.

Simple Parity-Check Code

Perhaps the most familiar error-detecting code is the simple parity-check code. In this code, a *k-bit* dataword is changed to an n-bit codeword where $n = k + 1$. The extra bit, called the parity bit, is selected to make the total number of Is in the codeword even. Although some implementations specify an odd number of Is, we discuss the even case. The minimum Hamming distance for this category is $d_{min} = 2$, which means that the code is a single-bit error-detecting code; it cannot correct any error.

> A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

Our first code (Table 10.1) is a parity-check code with $k = 2$ and $n = 3$. The code in Table 10.3 is also a parity-check code with $k = 4$ and $n = 5$.

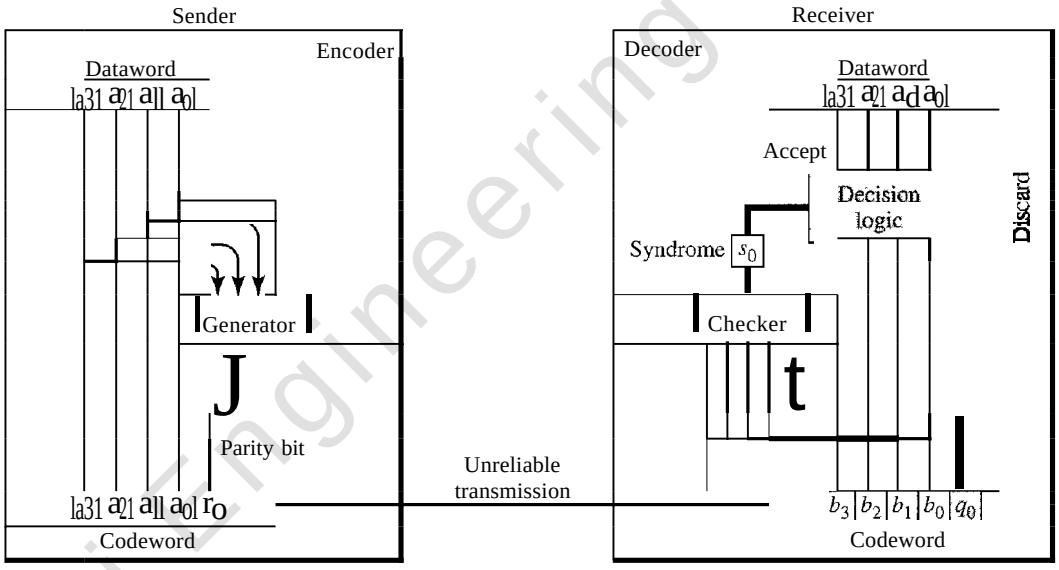
Figure 10.10 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver).

The encoder uses a generator that takes a copy of a 4-bit dataword *(ao,* aI' *a2'* and *a3)* and generates a parity bit *roo* The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of Is in the codeword even.

Table 10.3 *Simple parity-check code C(5, 4)*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 *Encoder and decoder for simple parity-check code*



This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

$$r_0 = a_3 + a_2 + a_1 + a_0 \qquad \text{(modulo-2)}$$

If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even.

The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the syndrome, is just 1 bit. The syndrome is 0 when the number of Is in the received codeword is even; otherwise, it is 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \qquad \text{(modulo-2)}$$

The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no error in the received codeword; the data portion of the received codeword is accepted as the dataword; if the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

Example 10.12

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.
2. One single-bit error changes *a1'* The received codeword is 10011. The syndrome is 1. No dataword is created.
3. One single-bit error changes *r00* The received codeword is 10110. The syndrome is 1. No dataword is created. Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.
4. An error changes *ro* and a second error changes *a3'* The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value. The simple parity-check decoder cannot detect an even number of errors. The errors cancel each other out and give the syndrome a value of 0.
5. Three bits-a3, *az,* and a1-are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

A simple parity-check code can detect an odd number of errors.

A better approach is the two-dimensional parity check. In this method, the dataword is organized in a table (rows and columns). In Figure 10.11, the data to be sent, five 7-bit bytes, are put in separate rows. For each row and each column, 1 parity-check bit is calculated. The whole table is then sent to the receiver, which finds the syndrome for each row and each column. As Figure 10.11 shows, the two-dimensional parity check can detect up to three errors that occur anywhere in the table (arrows point to the locations of the created nonzero syndromes). However, errors affecting 4 bits may not be detected.

Hamming Codes

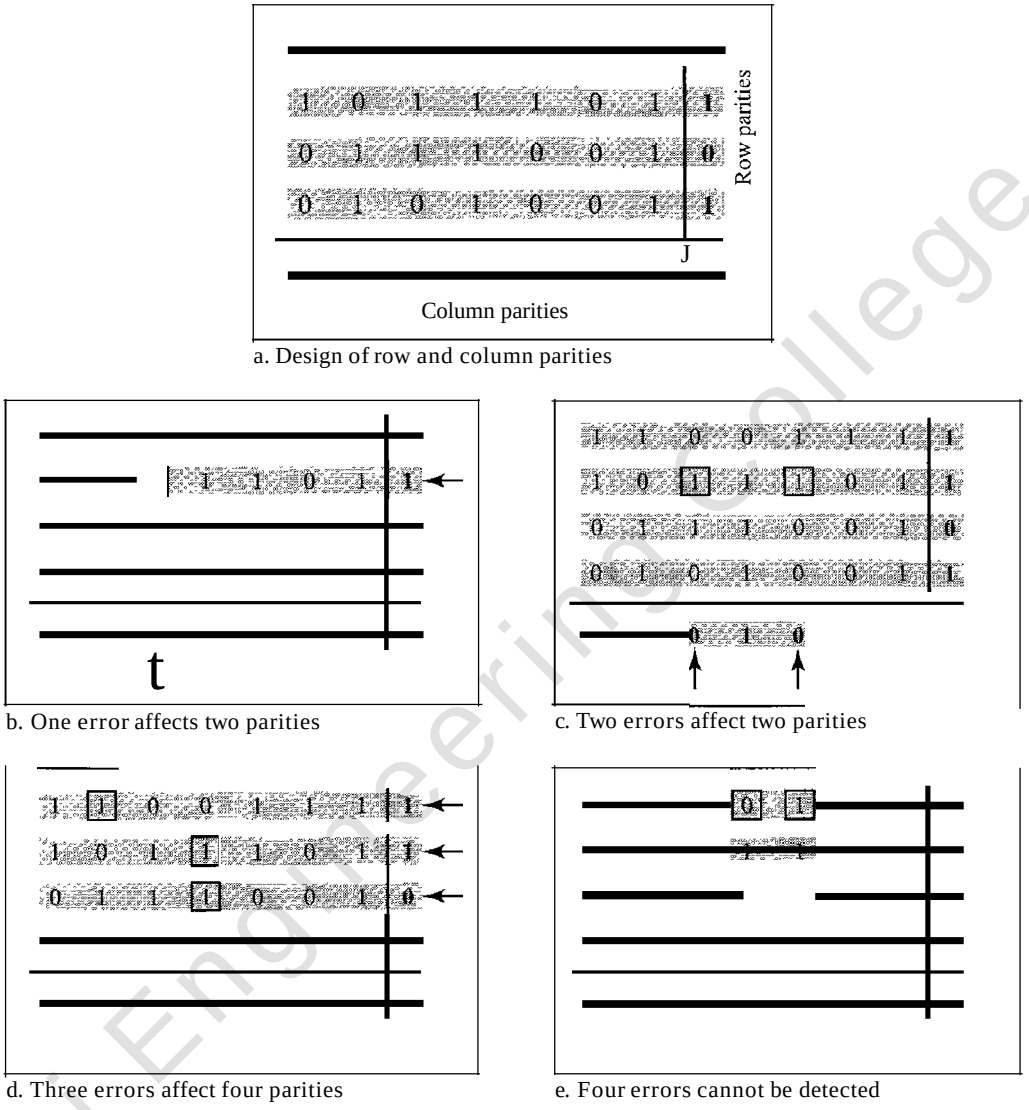
Now let us discuss a category of error-correcting codes called Hamming codes. These codes were originally designed with $d_{min} = 3$, which means that they can detect up to two errors or correct one single error. Although there are some Hamming codes that can correct more than one error, our discussion focuses on the single-bit error-correcting code.

First let us find the relationship between n and k in a Hamming code. We need to choose an integer $m >= 3$. The values of n and k are then calculated from mas $n = 2^m - 1$ and $k ::: n - m$. The number of check bits $r = m$.

All Hamming codes discussed in this book have $d_{min} = 3$.
The relationship between m and n in these codes is $n = 2^m - 1$.

For example, if $m = 3$, then $n ::: 7$ and $k::: 4$. This is a Hamming code $C(7, 4)$ with $d_{min} = 3$. Table 10.4 shows the datawords and codewords for this code.

Figure 10.11 *Two-dimensional parity-check code*



a. Design of row and column parities

b. One error affects two parities

c. Two errors affect two parities

d. Three errors affect four parities

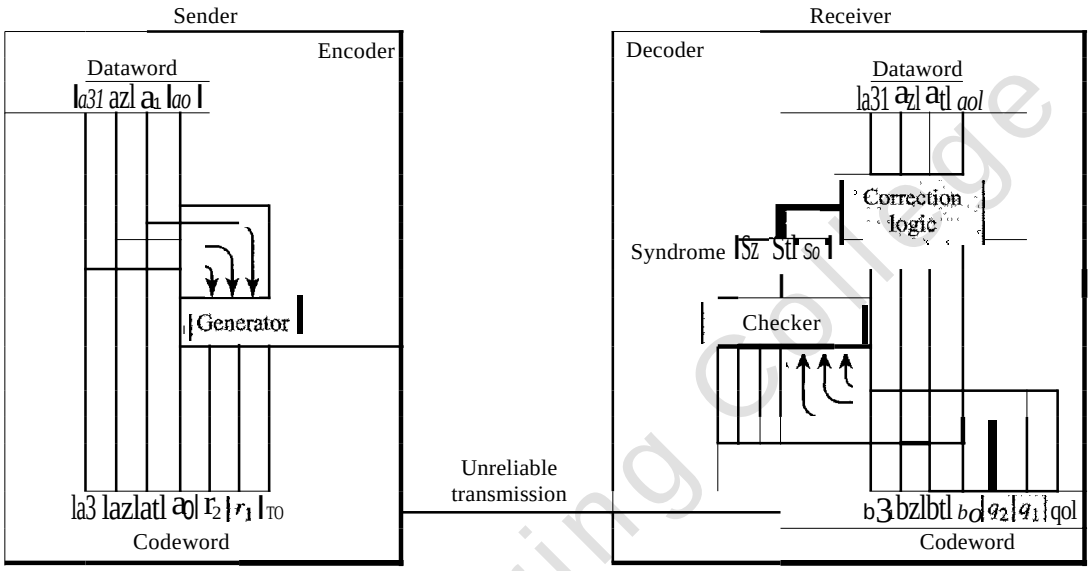
e. Four errors cannot be detected

Table 10.4 *Hamming code C(7, 4)*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011 00
0100	0100011	1100	1100101
0101	01011 10	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Figure 10.12 shows the structure of the encoder and decoder for this example.

Figure 10.12 *The structure of the encoder and decoder for a Hamming code*



A copy of a 4-bit dataword is fed into the generator that creates three parity checks *ro,* rI' and *r2'* as shown below:

$$r_0 = a_2 + a_1 + a_0 \quad \text{modulo-2}$$
$$r_1 = a_3 + a_2 + a_1 \quad \text{modulo-2}$$
$$r_2 = a_1 + a_0 + a_3 \quad \text{modulo-2}$$

In other words, each of the parity-check bits handles 3 out of the 4 bits of the dataword. The total number of 1s in each 4-bit combination (3 dataword bits and 1 parity bit) must be even. We are not saying that these three equations are unique; any three equations that involve 3 of the 4 bits in the dataword and create independent equations (a combination of two cannot create the third) are valid.

The checker in the decoder creates a 3-bit syndrome (s2s1s0) in which each bit is the parity check for 4 out of the 7 bits in the received codeword:

$$s_0 = b_2 + b_1 + b_0 + q_0 \quad \text{modulo-2}$$
$$s_1 = b_3 + b_2 + b_1 + q_1 \quad \text{modulo-2}$$
$$s_2 = b_1 + b_0 + b_3 + q_2 \quad \text{modulo-2}$$

The equations used by the checker are the same as those used by the generator with the parity-check bits added to the right-hand side of the equation. The 3-bit syndrome creates eight different bit patterns (000 to 111) that can represent eight different conditions. These conditions define a lack of error or an error in 1 of the 7 bits of the received codeword, as shown in Table 10.5.

Table 10.5 *Logical decision made by the correction logic analyzer a/the decoder*

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	ql	b_2	$q2$	b_O	b_3	b_l

Note that the generator is not concerned with the four cases shaded in Table 10.5 because there is either no error or an error in the parity bit. In the other four cases, 1 of the bits must be flipped (changed from 0 to 1 or 1 to 0) to find the correct dataword.

The syndrome values in Table 10.5 are based on the syndrome bit calculations. For example, if $q0$ is in error, $S0$ is the only bit affected; the syndrome, therefore, is 001. If b_2 is in error, $S0$ and $s1$ are the bits affected; the syndrome, therefore is OIl. Similarly, if b_l is in error, all 3 syndrome bits are affected and the syndrome is 111.

There are two points we need to emphasize here. First, if two errors occur during transmission, the created dataword might not be the right one. Second, if we want to use the above code for error detection, we need a different design.

Example 10.13

Let us trace the path of three datawords from the sender to the destination:

1. The dataword 0100 becomes the codeword 0100011. The codeword 01 00011 is received. The syndrome is 000 (no error), the final dataword is 0100.
2. The dataword 0111 becomes the codeword 0111001. The codeword 0011001 is received. The syndrome is 011. According to Table 10.5, b_2 is in error. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.
3. The dataword 1101 becomes the codeword 1101000. The codeword 0001000 is received (two errors). The syndrome is 101, which means that b_O is in error. After flipping b_O, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.

Example 10.14

We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

Solution

We need to make $k = n - m$ greater than or equal to 7, or $2^{in} - 1 - m \geq 7$.

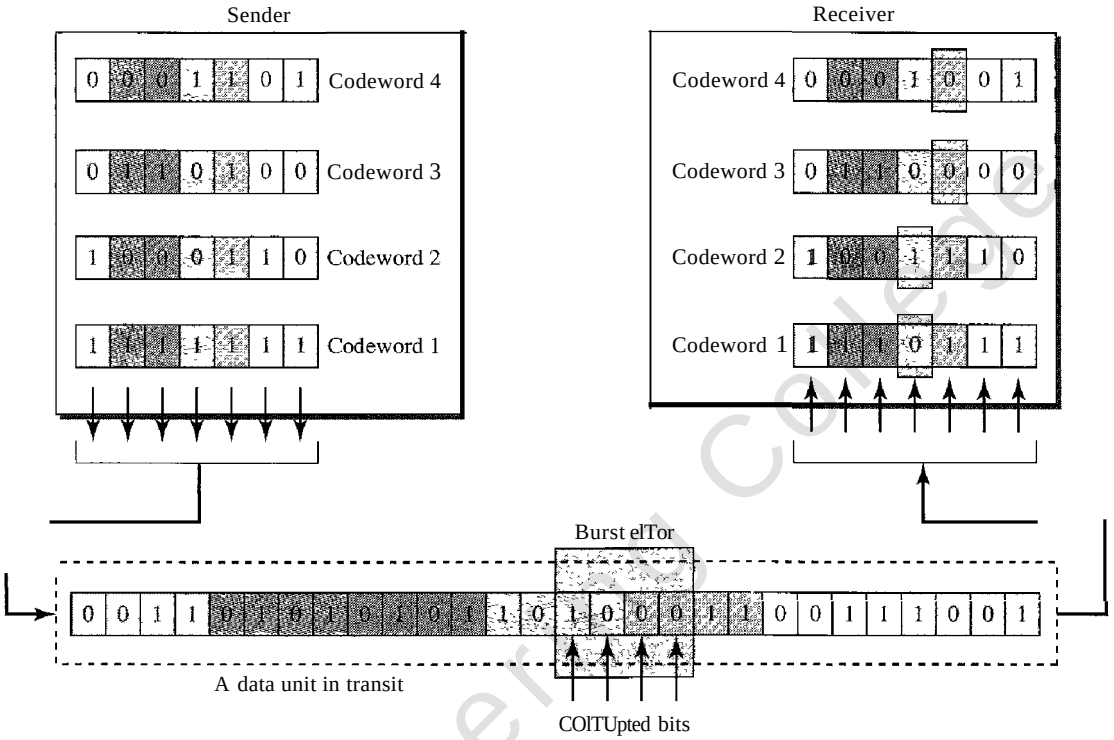
1. If we set $m = 3$, the result is $n = 2^3 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.
2. If we set $m = 4$, then $n = 2^4 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is C(l5, 11). There are methods to make the dataword a specific size, but the discussion and implementation are beyond the scope of this book.

Performance

A Hamming code can only correct a single error or detect a double error. However, there is a way to make it detect a burst error, as shown in Figure 10.13.

The key is to split a burst error between several codewords, one error for each codeword. In data communications, we normally send a packet or a frame of data. To make the Hamming code respond to a burst error of size N, we need to make N codewords out of our frame. Then, instead of sending one codeword at a time, we arrange the codewords in a table and send the bits in the table a column at a time. In Figure 10.13, the bits are sent colunm by column (from the left). In each column, the bits are sent from the bottom to the top. In this way, a frame is made out of the four codewords and sent to the receiver. Figure 10.13 shows

Figure 10.13 *Burst error correction using Hamming code*



that when a burst error of size 4 corrupts the frame, only 1 bit from each codeword is corrupted. The corrupted bit in each codeword can then easily be corrected at the receiver.

10.4 CYCLIC CODES

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word a_0 to a_6, and the bits in the second word b_0 to b_6, we can shift the bits by using the following:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a category of cyclic codes called the cyclic redundancy check (CRC) that is used in networks such as LANs and WANs.

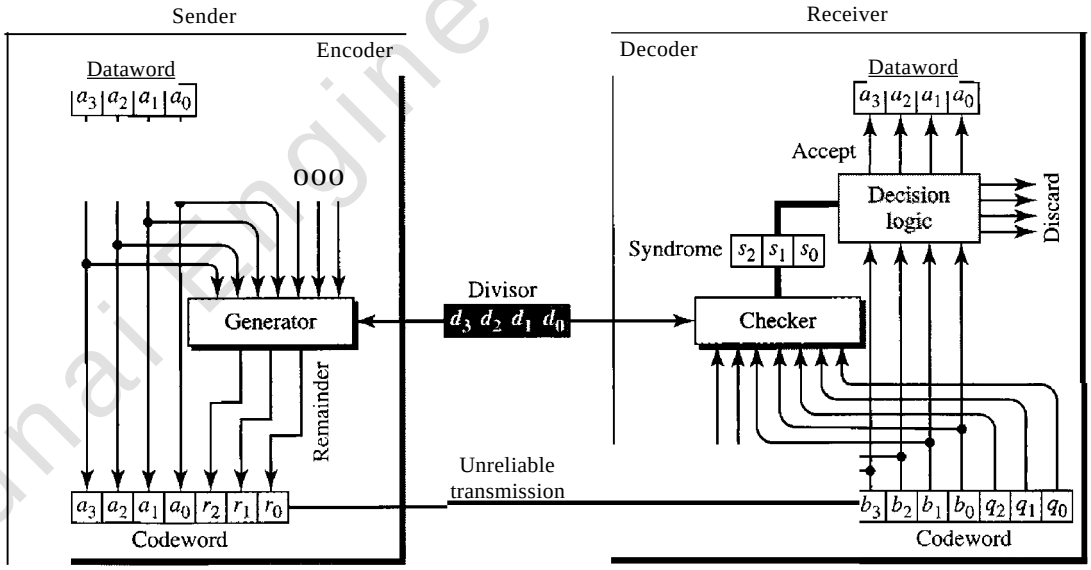
Table 10.6 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

Table 10.6 *A CRC code with C(7,* 4)

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 10.14 shows one possible design for the encoder and decoder.

Figure 10.14 *CRC encoder and decoder*



In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* - *k* (3 here) Os to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size *n* - *k* + I (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient ofthe division is discarded; the remainder *(r2rlro)* is appended to the dataword to create the codeword.

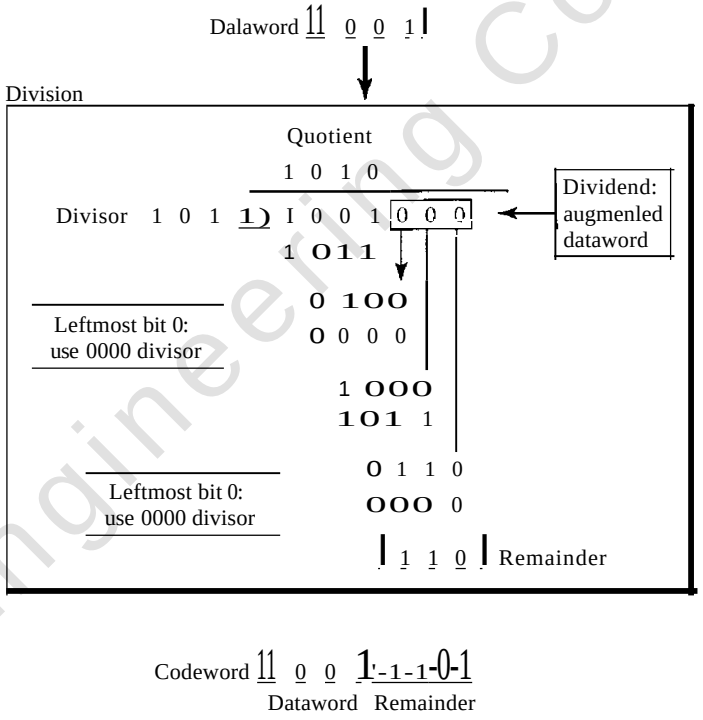
The decoder receives the possibly corrupted codeword. A copy of all *n* bits is fed to the checker which is a replica of the generator. The remainder produced by the checker

is a syndrome of $n - k$ (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes the dataword and augments it with $n - k$ number of as. It then divides the augmented dataword by the divisor, as shown in Figure 10.15.

Figure 10.15 *Division in CRC encoder*



The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. However, as mentioned at the beginning of the chapter, in this case addition and subtraction are the same. We use the XOR operation to do both.

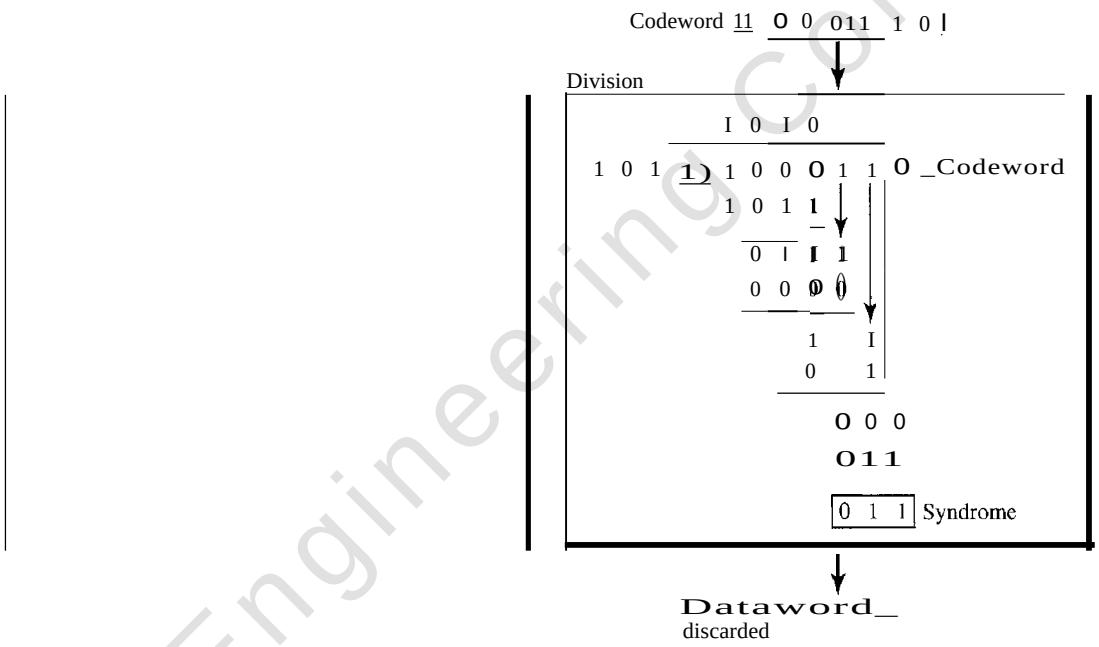
As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-Os divisor.

When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits *(r2' r1'* and *ro).* They are appended to the dataword to create the codeword.

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. **If** the syndrome is all Os, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.16 shows two cases: The left-hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all Os (it is OIl).

Figure 10.16 *Division in the CRC decoder for two cases*



Divisor

You may be wondering how the divisor] 011 is chosen. Later in the chapter we present some criteria, but in general it involves abstract algebra.

Hardware Implementation

One of the advantages of a cyclic code is that the encoder and decoder can easily and cheaply be implemented in hardware by using a handful of electronic devices. Also, a hardware implementation increases the rate of check bit and syndrome bit calculation. **In** this section, we try to show, step by step, the process. The section, however, is optional and does not affect the understanding of the rest of the chapter.

Divisor

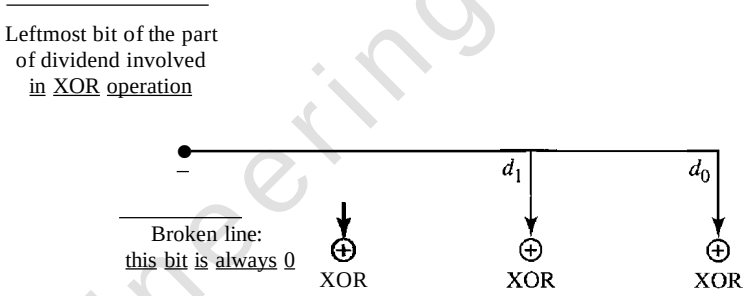
Let us first consider the divisor. We need to note the following points:

1. The divisor is repeatedly XORed with part of the dividend.

2. The divisor has $n - k + 1$ bits which either are predefined or are all 0s. In other words, the bits do not change from one dataword to another. In our previous example, the divisor bits were either 1011 or 0000. The choice was based on the leftmost bit of the part of the augmented data bits that are active in the XOR operation.
3. A close look shows that only $n - k$ bits of the divisor is needed in the XOR operation. The leftmost bit is not needed because the result of the operation is always 0, no matter what the value of this bit. The reason is that the inputs to this XOR operation are either both 0s or both 1s. In our previous example, only 3 bits, not 4, is actually used in the XOR operation.

Using these points, we can make a fixed (hardwired) divisor that can be used for a cyclic code if we know the divisor pattern. Figure 10.17 shows such a design for our previous example. We have also shown the XOR devices used for the operation.

Figure 10.17 *Hardwired design of the divisor in CRC*



Note that if the leftmost bit of the part of dividend to be used in this step is 1, the divisor bits $(d_2 d_1 d_0)$ are all; if the leftmost bit is 0, the divisor bits arc 000. The design provides the right choice based on the leftmost bit.

Augmented Dataword

In our paper-and-pencil division process in Figure 10.15, we show the augmented dataword as fixed in position with the divisor bits shifting to the right, 1 bit in each step. The divisor bits are aligned with the appropriate part of the augmented dataword. Now that our divisor is fixed, we need instead to shift the bits of the augmented dataword to the left (opposite direction) to align the divisor bits with the appropriate part. There is no need to store the augmented dataword bits.

Remainder

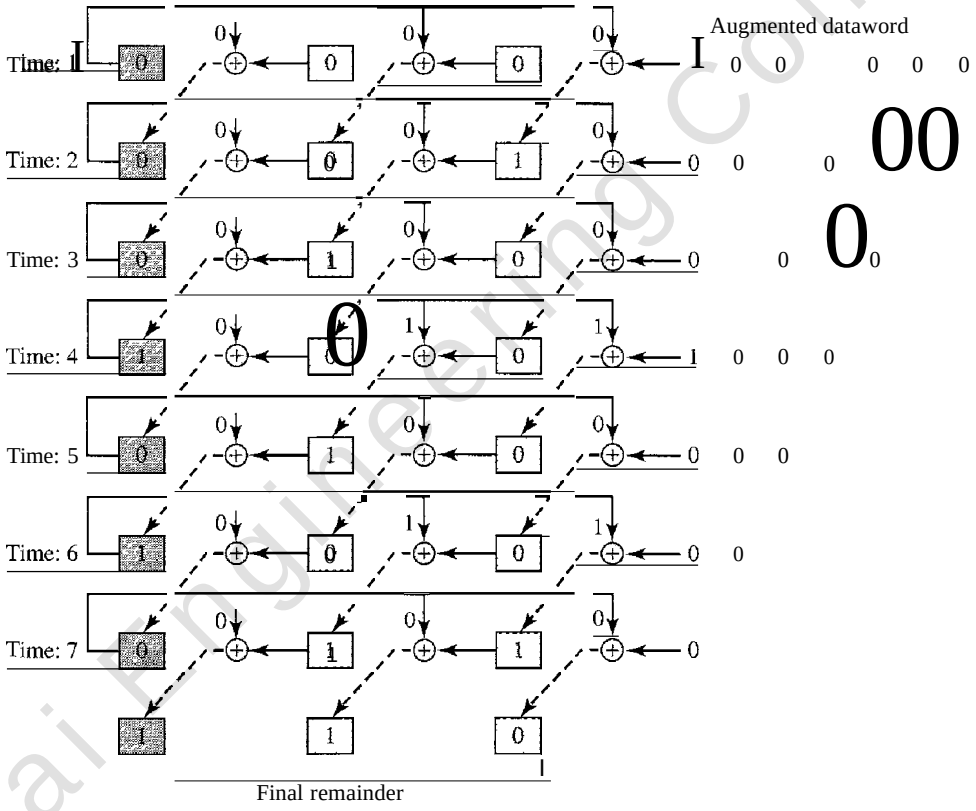
In our previous example, the remainder is 3 bits $(n - k$ bits in general) in length. We can use three registers (single-bit storage devices) to hold these bits. To find the final remainder of the division, we need to modify our division process. The following is the step-by-step process that can be used to simulate the division process in hardware (or even in software).

1. We assume that the remainder is originally all 0s (000 in our example).

2. At each time click (arrival of 1 bit from an augmented dataword), we repeat the following two actions:
   a. We use the leftmost bit to make a decision about the divisor (011 or 000).
   b. The other 2 bits of the remainder and the next bit from the augmented dataword (total of 3 bits) are XORed with the 3-bit divisor to create the next remainder.

Figure 10.18 shows this simulator, but note that this is not the final design; there will be more improvements.

Figure 10.18 *Simulation of division in CRC encoder*

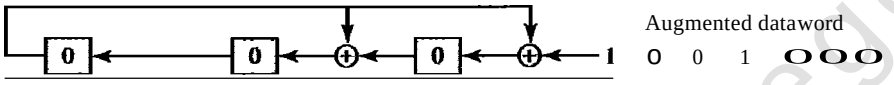


At each clock tick, shown as different times, one of the bits from the augmented dataword is used in the XOR process. If we look carefully at the design, we have seven steps here, while in the paper-and-pencil method we had only four steps. The first three steps have been added here to make each step equal and to make the design for each step the same. Steps 1, 2, and 3 push the first 3 bits to the remainder registers; steps 4, 5, 6, and 7 match the paper-and-pencil design. Note that the values in the remainder register in steps 4 to 7 exactly match the values in the paper-and-pencil design. The final remainder is also the same.

The above design is for demonstration purposes only. It needs simplification to be practical. First, we do not need to keep the intermediate values of the remainder bits; we need only the final bits. We therefore need only 3 registers instead of 24. After the XOR operations, we do not need the bit values of the previous remainder. Also, we do

not need 21 XOR devices; two are enough because the output of an XOR operation in which one of the bits is 0 is simply the value of the other bit. This other bit can be used as the output. With these two modifications, the design becomes tremendously simpler and less expensive, as shown in Figure 10.19.

Figure 10.19 *The CRC encoder design using shift registers*

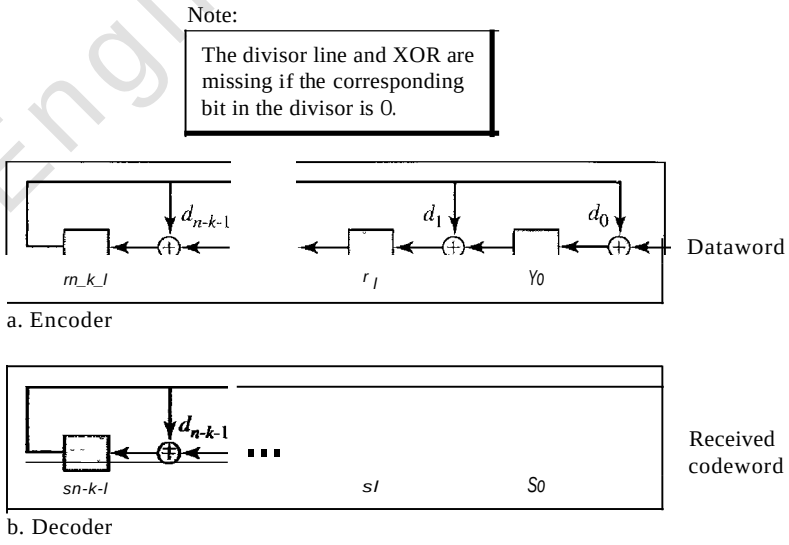


We need, however, to make the registers shift registers. A I-bit shift register holds a bit for a duration of one clock time. At a time click, the shift register accepts the bit at its input port, stores the new bit, and displays it on the output port. The content and the output remain the same until the next input arrives. When we connect several I-bit shift registers together, it looks as if the contents of the register are shifting.

General Design

A general design for the encoder and decoder is shown in Figure 10.20.

Figure 10.20 *General design of encoder and decoder of a CRC code*



Note that we have $n - k$ I-bit shift registers in both the encoder and decoder. We have up to $n - k$ XOR devices, but the divisors normally have several 0s in their pattern, which reduces the number of devices. Also note that, instead of augmented datawords, we show the dataword itself as the input because after the bits in the dataword are all fed into the encoder, the extra bits, which all are 0s, do not have any effect on the right-most XOR. Of course, the process needs to be continued for another $n - k$ steps before

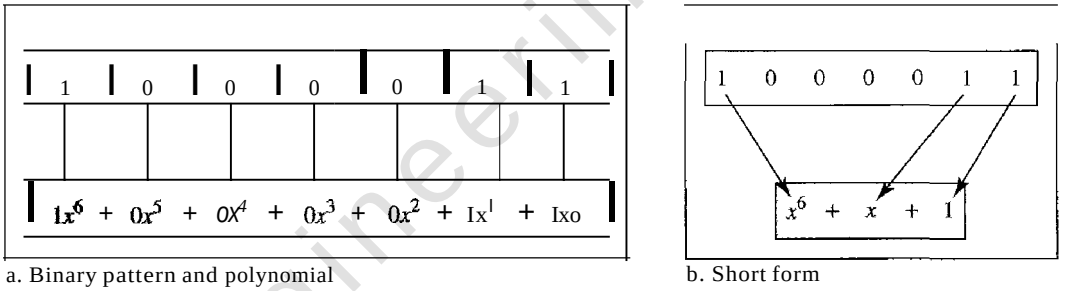
the check bits are ready. This fact is one of the criticisms of this design. Better schemes have been designed to eliminate this waiting time (the check bits are ready after *k* steps), but we leave this as a research topic for the reader. In the decoder, however, the entire codeword must be fed to the decoder before the syndrome is ready.

Polynomials

A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. Again, this section is optional.

A pattern of 0s and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Figure 10.21 shows a binary pattern and its polynomial representation. In Figure 10.21a we show how to translate a binary pattern to a polynomial; in Figure 1O.21b we show how the polynomial can be shortened by removing all terms with zero coefficients and replacing $\times 1$ by x and xO by 1.

Figure 10.21 *A polynomial to represent a binary word*



a. Binary pattern and polynomial

b. Short form

Figure 10.21 shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x^{23} + x^3 + 1$. Here the bit pattern is 24 bits in length (three Is and twenty-one Os) while the polynomial is just three terms.

Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x^6 + x + 1$ is 6. Note that the degree of a polynomial is 1 less that the number of bits in the pattern. The bit pattern in this case has 7 bits.

Adding and Subtracting Polynomials

Adding and subtracting polynomials in mathematics are done by adding or subtracting the coefficients of terms with the same power. In our case, the coefficients are only 0 and 1, and adding is in modulo-2. This has two consequences. First, addition and subtraction are the same. Second, adding or subtracting is done by combining terms and deleting pairs of identical terms. For example, adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ gives just $x^6 + x^5$. The terms x^4 and x^2 are deleted. However, note that if we add, for example, three polynomials and we get x^2 three times, we delete a pair of them and keep the third.

Multiplying or Dividing Terms

In this arithmetic, multiplying a term by another term is very simple; we just add the powers. For example, x^3 x x^4 is x^7. For dividing, we just subtract the power of the second term from the power of the first. For example, x^5 / x^2 is x^3.

Multiplying Two Polynomials

Multiplying a polynomial by another is done term by term. Each term of the first polynomial must be multiplied by all terms of the second. The result, of course, is then simplified, and pairs of equal terms are deleted. The following is an example:

$$\begin{aligned}(x^5 + x^3 + x^2 + x)(x^2 + x + 1) \\ &= x^7 + x^6 + x^5 + x^5 + x^4 + x^3 + x^4 + x^3 + x^2 + x^3 + x^2 + x \\ &= x^7 + x^6 + x^3 + x\end{aligned}$$

Dividing One Polynomial by Another

Division of polynomials is conceptually the same as the binary division we discussed for an encoder. We divide the first term of the dividend by the first term of the divisor to get the first term of the quotient. We multiply the term in the quotient by the divisor and subtract the result from the dividend. We repeat the process until the dividend degree is less than the divisor degree. We will show an example of division later in this chapter.

Shifting

A binary pattern is often shifted a number of bits to the right or left. Shifting to the left means adding extra 0s as rightmost bits; shifting to the right means deleting some rightmost bits. Shifting to the left is accomplished by multiplying each term of the polynomial by x^m, where m is the number of shifted bits; shifting to the right is accomplished by dividing each term of the polynomial by x^m. The following shows shifting to the left and to the right. Note that we do not have negative powers in the polynomial representation.

Shifting left 3 bits:	10011 becomes 10011000	$x^4 + x + 1$ becomes $x^7 + x^4 + x^3$
Shifting right 3 bits:	10011 becomes 10	$x^4 + x + 1$ becomes x

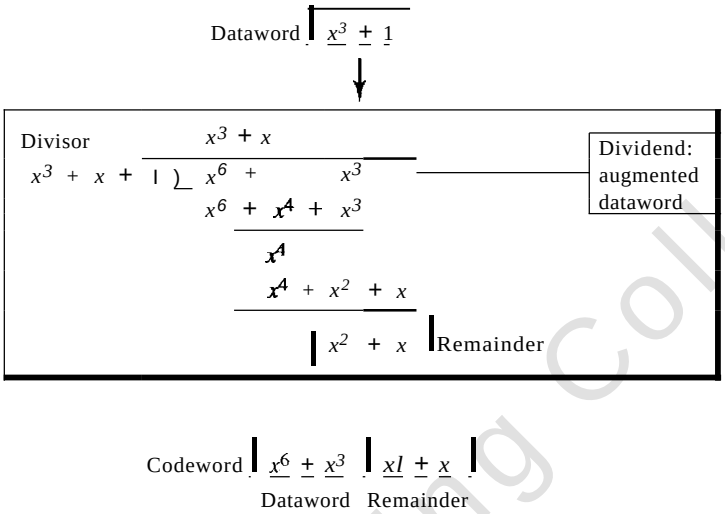
When we augmented the dataword in the encoder of Figure 10.15, we actually shifted the bits to the left. Also note that when we concatenate two bit patterns, we shift the first polynomial to the left and then add the second polynomial.

Cyclic Code Encoder Using Polynomials

Now that we have discussed operations on polynomials, we show the creation of a codeword from a dataword. Figure 10.22 is the polynomial version of Figure 10.15. We can see that the process is shorter. The dataword 1001 is represented as $x^3 + 1$. The divisor 1011 is represented as $x^3 + x + 1$. To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x^3). The result is $x^6 + x^3$. Division is straightforward. We divide the first term of the dividend, x^6, by the first term of the divisor, x^3. The first term of the quotient is then x^6/x^3, or x^3. Then we multiply x^3 by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend. The

result is x^4, with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.

Figure 10.22 *CRC division using polynomials*



It can be seen that the polynomial representation can easily simplify the operation of division in this case, because the two steps involving all-Os divisors are not needed here. (Of course, one could argue that the all-Os divisor step can also be eliminated in binary division.) In a polynomial representation, the divisor is normally referred to as the generator polynomial *t(x).*

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

Cyclic Code Analysis

We can analyze a cyclic code to find its capabilities by using polynomials. We define the following, *wheref(x)* is a polynomial with binary coefficients.

Dataword: *d(x)* Codeword: *c(x)* Generator: *g(x)*
Syndrome: *sex)* Error: *e(x)*

If *sex)* is not zero, then one or more bits is corrupted. However, if *sex)* is zero, either no bit is corrupted or the decoder failed to detect any errors.

In a cyclic code,

1. If $s(x) \neq 0$, one or more bits is corrupted.
2. If *sex)* $=0$, either
   a. No bit is corrupted. or
   b. Some bits are corrupted, but the decoder failed to detect them.

In our analysis we want to find the criteria that must be imposed on the generator, $g(x)$ to detect the type of error we especially want to be detected. Let us first find the relationship among the sent codeword, error, received codeword, and the generator. We can say

$$\text{Received codeword} = c(x) + e(x)$$

In other words, the received codeword is the sum of the sent codeword and the error. The receiver divides the received codeword by $g(x)$ to get the syndrome. We can write this as

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

The first term at the right-hand side of the equality does not have a remainder (according to the definition of codeword). So the syndrome is actually the remainder of the second term on the right-hand side. If this term does not have a remainder (syndrome = 0), either $e(x)$ is 0 or $e(x)$ is divisible by $g(x)$. We do not have to worry about the first case (there is no error); the second case is very important. Those errors that are divisible by $g(x)$ are not caught.

In a cyclic code, those $e(x)$ errors that are divisible by $g(x)$ are not caught.

Let us show some specific errors and see how they can be caught by a well-designed $g(x)$.

Single-Bit Error

What should be the structure of $g(x)$ to guarantee the detection of a single-bit error? A single-bit error is $e(x) = x^i$, where i is the position of the bit. If a single-bit error is caught, then x^i is not divisible by $g(x)$. (Note that when we say *not divisible,* we mean that there is a remainder.) If $g(x)$ has at least two terms (which is normally the case) and the coefficient of x^0 is not zero (the rightmost bit is 1), then $e(x)$ cannot be divided by $g(x)$.

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

Example 10.15

Which of the following $g(x)$ values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$
b. x^3
c. 1

Solution

a. No xi can be divisible by $x + 1$. In other words, $xi/ex + 1)$ always has a remainder. So the syndrome is nonzero. Any single-bit error can be caught.

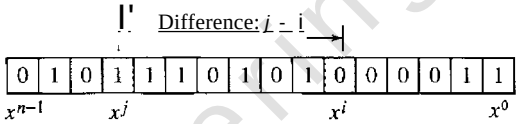
b. If i is equal to or greater than 3, xi is divisible by $g(x)$. The remainder of x^i/x^3 is zero, and the receiver is fooled into believing that there is no error, although there might be one. Note that in this case, the corrupted bit must be in position 4 or above. All single-bit errors in positions I to 3 are caught.

c. All values of i make x^i divisible by $g(x)$. No single-bit error can be caught. In addition, this $g(x)$ is useless because it means the codeword is just the dataword augmented with $n - k$ zeros.

Two Isolated Single-Bit Errors

Now imagine there are two single-bit isolated errors. Under what conditions can this type of error be caught? We can show this type of error as $e(x) = x^j + xi$. The values of i and j define the positions of the errors, and the difference $j - i$ defines the distance between the two errors, as shown in Figure 10.23.

Figure 10.23 *Representation o/two isolated single-bit errors using polynomials*



We can write $e(x) = x^i(x^{j-i} + 1)$. If $g(x)$ has more than one term and one term is xo, it cannot divide x^i, as we saw in the previous section. So if $g(x)$ is to divide $e(x)$, it must divide $x^{j-i} + 1$. In other words, $g(x)$ must not divide $x^t + 1$, where t is between 0 and $n - 1$. However, $t = 0$ is meaningless and $t =$ I is needed as we will see later. This means t should be between $2 and n-1$.

If a generator cannot divide $x^t + 1$ (t between 0 and $n - 1$), then all isolated double errors can be detected.

Example 10.16

Find the status of the following generators related to two isolated, single-bit errors.

a. $x + 1$

b. $x^4 + I$

c. $x^7 + x^6 + 1$

d. $x^{Is} + x^{I4} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. This generator cannot detect two errors that are four positions apart. The two errors can be anywhere, but if their distance is 4, they remain undetected.

c. This is a good choice for this purpose.

d. This polynomial cannot divide any error of type $x^t + 1$ if t is less than 32,768. This means that a codeword with two isolated errors that are next to each other or up to 32,768 bits apart can be detected by this generator.

Odd Numbers of Errors

A generator with a factor of $x + 1$ can catch all odd numbers of errors. This means that we need to make $x + 1$ a factor of any generator. Note that we are not saying that the generator itself should be $x + 1$; we are saying that it should have a factor of $x + 1$. If it is only $x + 1$, it cannot catch the two adjacent isolated errors (see the previous section). For example, $x^4 + x^2 + x + 1$ can catch all odd-numbered errors since it can be written as a product of the two polynomials $x + 1$ and $x^3 + x^2 + 1$.

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

Burst Errors

Now let us extend our analysis to the burst error, which is the most important of all. A burst error is of the form $e(x) = (x^j + \ldots + xi)$. Note the difference between a burst error and two isolated single-bit errors. The first can have two terms or more; the second can only have two terms. We can factor out xi and write the error as $x^i(x^{j-i} + \ldots + 1)$. If our generator can detect a single error (minimum condition for a generator), then it cannot divide xi. What we should worry about are those generators that divide $x^{j-i} + \ldots + 1$. In other words, the remainder of $(x^{j-i} + \ldots + 1)/(x^r + \ldots + 1)$ must not be zero. Note that the denominator is the generator polynomial. We can have three cases:

1. If $j - i < r$, the remainder can never be zero. We can write $j - i = L - 1$, where L is the length of the error. So $L - 1 < r$ or $L < r + 1$ or $L \leq r$. This means all burst errors with length smaller than or equal to the number of check bits r will be detected.
2. In some rare cases, if $j - i = r$, or $L = r + 1$, the syndrome is 0 and the error is undetected. It can be proved that in these cases, the probability of undetected burst error of length $r + 1$ is $(1/2)^{r-1}$. For example, if our generator is $x^{14} + x^3 + 1$, in which $r = 14$, a burst error of length $L = 15$ can slip by undetected with the probability of (1/2)14-1 or almost 1 in 10,000.
3. In some rare cases, if $j - i > r$, or $L > r + 1$, the syndrome is 0 and the error is undetected. It can be proved that in these cases, the probability of undetected burst error of length greater than $r + 1$ is $(1/2)^r$. For example, if our generator is $x^{14} + x^3 + 1$, in which $r = 14$, a burst error of length greater than 15 can slip by undetected with the probability of (112)14 or almost 1 in 16,000 cases.

- All burst errors with $L \leq$ r will be detected.
- All burst errors with $L =$ r + 1 will be detected with probability 1 - *(112/-1*.
- All burst errors with $L >$ r + 1 will be detected with probability 1 – (1/2[.

Example 10.17

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$

b. $x^{18} + x^7 + x + 1$

c. $x^{32} + x^{23} + x^7 + 1$

Solution

a. This generator can detect all burst errors with a length less than or equal to 6 bits; 3 out of 100 burst errors with length 7 will slip by; 16 out of 1000 burst errors of length 8 or more will slip by.
b. This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of I million burst errors of length 20 or more will slip by.
c. This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.

Summary

We can summarize the criteria for a good polynomial generator:

A good polynomial generator needs to have the following characteristics:

1. It should have at least two terms.
2. The coefficient of the term xO should be 1.
3. It should not divide $x^t + 1$, for t between 2 and $n - 1$.
4. It should have the factor $x + 1$.

Standard Polynomials

Some standard polynomials used by popular protocols for eRe generation are shown in Table 10.7.

Table 10.7 *Standard polynomials*

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-lO	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATMAAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Advantages of Cyclic Codes

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware. This has made cyclic codes a good candidate for many networks.

Other Cyclic Codes

The cyclic codes we have discussed in this section are very simple. The check bits and syndromes can be calculated by simple algebra. There are, however, more powerful polynomials that are based on abstract algebra involving Galois fields. These are beyond

the scope of this book. One of the most interesting of these codes is the Reed-Solomon code used today for both detection and correction.

10.5 CHECKSUM

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking.

Like linear and cyclic codes, the checksum is based on the concept of redundancy. Several protocols still use the checksum for error detection as we will see in future chapters, although the tendency is to replace it with a CRe. This means that the CRC is also used in layers other than the data link layer.

Idea

The concept of the checksum is not difficult. Let us illustrate it with a few examples.

Example 10.18

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12,0,6,36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example 10.19

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the *checksum.* In this case, we send (7, 11, 12,0,6, -36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

The previous example has one major drawback. All of our data can be written as a 4-bit word (they are less than 15) except for the checksum. One solution is to use one's complement arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^n - 1$ using only n bits.† If the number has more than n bits, the extra leftmost bits need to be added to the n rightmost bits (wrapping). In one's complement arithmetic, a negative number can be represented by inverting all bits (changing a 0 to a 1 and a 1 to a 0). This is the same as subtracting the number from $2^n - 1$.

Example 10.20

How can we represent the number 21 in one's complement arithmetic using only four bits?

†Although one's complement can represent both positive and negative numbers, we are concerned only with unsigned representation here.

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

Example 10.21

How can we represent the number -6 in one's complement arithmetic using only four bits?

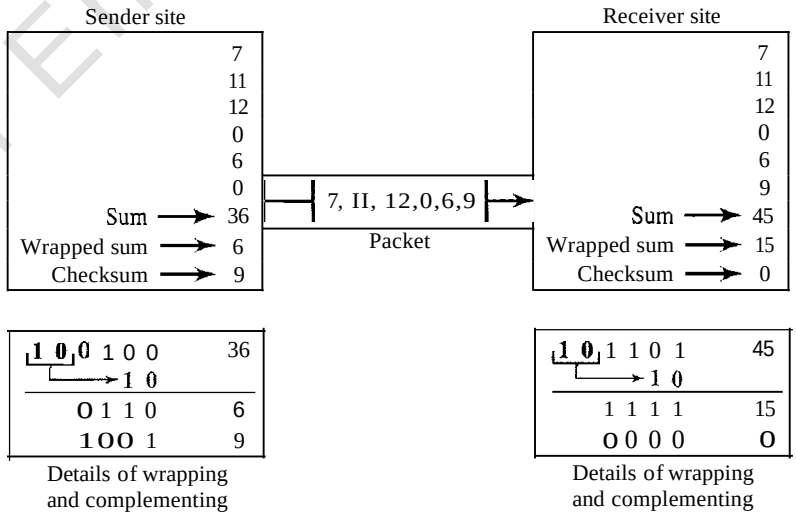
Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ (16 - 1 in this case).

Example 10.22

Let us redo Exercise 10.19 using one's complement arithmetic. Figure 10.24 shows the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sum is then complemented, resulting in the checksum value 9 (15 - 6 = 9). The sender now sends six data items to the receiver including the checksum 9. The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Figure 10.24



Internet Checksum

Traditionally, the Internet has been using a 16-bit checksum. The sender calculates the checksum by following these steps.

Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

The receiver uses the following steps for error detection.

Receiver site:

1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

The nature of the checksum (treating words as numbers and adding and complementing them) is well-suited for software implementation. Short programs can be written to calculate the checksum at the receiver site or to check the validity of the message at the receiver site.

Example 10.23

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (l6-bit) words. We use ASCII (see Appendix A) to change each byte to a 2-digit hexadecimal number. For example, F is represented as Ox46 and o is represented as Ox6F. Figure 10.25 shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is Ox36. We keep the rightmost digit (6) and insert the

Figure 10.25

1	0	1	3		Carries
4	6	6	F		(Fo)
7	2	6	F		(ro)
7	5	7	A		luz)
6	1	6	E		(an)
0	0	0	0		Checksum (initial)
8	F	C	6		Sum (partial)
			1		
8	F	C	7		Sum
7	0	3	8		Checksum (to send)

a. Checksum at the sender site

1	0	1	3		Carries
4	6	6	F		IFo)
7	2	6	F		(ro)
7	5	7	A		(uz)
6	1	6	E		(an)
7	0	3	8		Checksum (received)
F	F	F	E		Sum (partial)
			1		
F	F	F	F		Sum
0	0	0	0		Checksum (new}

b. Checksum at the receiver site

leftmost dight (3) as the carry in the second column. The process is repeated for each column. Hexadecimal numbers are reviewed in Appendix B.

Note that if there is any corruption, the checksum recalculated by the receiver is not all as. We leave this an exercise.

Performance

The traditional checksum uses a small number of bits (16) to detect errors in a message of any size (sometimes thousands of bits). However, it is not as strong as the CRC in error-checking capability. For example, if the value of one word is incremented and the value of another word is decremented by the same amount, the two errors cannot be detected because the sum and checksum remain the same. Also if the values of several words are incremented but the total change is a multiple of 65535, the sum and the checksum does not change, which means the errors are not detected. Fletcher and Adler have proposed some weighted checksums, in which each word is multiplied by a number (its weight) that is related to its position in the text. This will eliminate the first problem we mentioned. However, the tendency in the Internet, particularly in designing new protocols, is to replace the checksum with a CRC.

10.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Several excellent book are devoted to error coding. Among them we recommend [Ham80], [Zar02], [Ror96], and [SWE04].

RFCs

A discussion of the use of the checksum in the Internet can be found in RFC 1141.

10.7 KEY TERMS

block code
burst error
check bit
checksum
codeword
convolution code
cyclic code
cyclic redundancy check (CRC)
dataword
error
error correction
error detection
forward error correction
generator polynomial
Hamming code
Hamming distance
interference
linear block code
minimum Hamming distance
modular arithmetic

modulus
one's complement
parity bit
parity-check code
polynomial
redundancy
Reed-Solomon
register
retransmission
shift register
single-bit error
syndrome
two-dimensional parity check

10.8 SUMMARY

- Data can be corrupted during transmission. Some applications require that errors be detected and corrected.
- In a single-bit error, only one bit in the data unit has changed. A burst error means that two or more bits in the data unit have changed.
- To detect or correct errors, we need to send extra (redundant) bits with data.
- There are two main methods of error correction: forward error correction and correction by retransmission.
- We can divide coding schemes into two broad categories: *block coding* and *convolution* coding.
- In coding, we need to use modulo-2 arithmetic. Operations in this arithmetic are very simple; addition and subtraction give the same results. we use the XOR (exclusive OR) operation for both addition and subtraction.
- In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called codewords.
- In block coding, errors be detected by using the following two conditions:
  a. The receiver has (or can find) a list of valid codewords.
  b. The original codeword has changed to an invalid one.
- The Hamming distance between two words is the number of differences between corresponding bits. The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.
- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$. To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.
- In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.
- A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$. A simple parity-check code can detect an odd number of errors.
- All Hamming codes discussed in this book have $d_{min} = 3$. The relationship between m and n in these codes is $n = 2^m - 1$.
- Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

- A category of cyclic codes called the cyclic redundancy check (CRC) is used in networks such as LANs and WANs.
- A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1.
- Traditionally, the Internet has been using a 16-bit checksum, which uses *one's complement* arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^n - 1$ using only n bits.

10.9 PRACTICE SET

Review Questions

1. How does a single-bit error differ from a burst error?
2. Discuss the concept of redundancy in error detection and correction.
3. Distinguish between forward error correction versus error correction by retransmission.
4. What is the definition of a linear block code? What is the definition of a cyclic code?
5. What is the Hamming distance? What is the minimum Hamming distance?
6. How is the simple parity check related to the two-dimensional parity check?
7. In CRC, show the relationship between the following entities (size means the number of bits):
   a. The size of the dataword and the size of the codeword
   b. The size of the divisor and the remainder
   c. The degree of the polynomial generator and the size of the divisor
   d. The degree of the polynomial generator and the size of the remainder
8. What kind of arithmetic is used to add data items in checksum calculation?
9. What kind of error is undetectable by the checksum?
10. Can the value of a checksum be all 0s (in binary)? Defend your answer. Can the value be all1s (in binary)? Defend your answer.

Exercises

11. What is the maximum effect of a 2-ms burst of noise on data transmitted at the following rates?
    a. 1500 bps
    b. 12 kbps
    c. 100 kbps
    d. 100 Mbps
12. Apply the exclusive-or operation on the following pair of patterns (the symbol $\oplus$ means XOR):
    a. (10001) $\oplus$ (10000)
    b. (10001) $\oplus$ (10001) (What do you infer from the result?)
    c. (11100) $\oplus$ (00000) (What do you infer from the result?)
    d. (10011) $\oplus$ (11111) (What do you infer from the result?)

13. In Table 10.1, the sender sends dataword 10. A 3-bit burst error corrupts the codeword. Can the receiver detect the error? Defend your answer.
14. In Table 10.2, the sender sends dataword 10. If a 3-bit burst error corrupts the first three bits of the codeword, can the receiver detect the error? Defend your answer.
15. What is the Hamming distance for each of the following codewords:
    a. d (10000, 00000)
    b. d (10101, 10000)
    c. d (11111,11111)
    d. d (000, 000)
16. Find the minimum Hamming distance for the following cases:
    a. Detection of two errors.
    b. Correction of two errors.
    c. Detection of 3 errors or correction of 2 errors.
    d. Detection of 6 errors or correction of 2 errors.
17. Using the code in Table 10.2, what is the dataword if one of the following codewords is received?
    a. 01011
    b. 11111
    c. 00000
    d. 11011
18. Prove that the code represented by Table 10.8 is not a linear code. You need to find only one case that violates the linearity.

Table 10.8 *Table for Exercise 18*

Dataword	*Codeword*
00	00000
01	01011
10	10111
11	11111

19. Although it can mathematically be proved that a simple parity check code is a linear code, use manual testing of linearity for five pairs of the codewords in Table 10.3 to partially prove this fact.
20. Show that the Hamming code C(7,4) of Table lOA can detect two-bit errors but not necessarily three-bit error by testing the code in the following cases. The character "∇" in the burst error means no error; the character "E" means an error.
    a. Dataword: 0100 Burst error: ∇EE∇∇∇∇
    b. Dataword: 0111 Burst error: E∇∇∇∇∇E
    c. Dataword: 1111 Burst error: E∇E∇∇∇E
    d. Dataword: 0000 Burst error: EE∇E∇∇∇

21. Show that the Hamming code C(7,4) of Table lOA can correct one-bit errors but not more by testing the code in the following cases. The character "∇" in the burst error means no error; the character "E" means an error.
    a. Dataword: 0100 Burst error: E∇∇∇∇∇∇
    b. Dataword: 0111 Burst error: ∇E∇∇∇∇∇
    c. Dataword: 1111 Burst error: E∇∇∇∇∇E
    d. Dataword: 0000 Burst error: EE∇∇∇∇E
22. Although it can be proved that code in Table 10.6 is both linear and cyclic, use only two tests to partially prove the fact:
    a. Test the cyclic property on codeword 0101100.
    b. Test the linear property on codewords 0010110 and 1111111.
23. We need a dataword of at least 11 bits. Find the values of k and n in the Hamming code $C(n, k)$ with $d_{min} = 3$.
24. Apply the following operations on the corresponding polynomials:
    a. $(x^3 + xl + x + 1) + (x^4 + xl + x + 1)$
    b. $(x^3 + xl + x + 1) - (x^4 + xl + x + 1)$
    c. $(x^3 + xl) \times (x^4 + x^2 + x + 1)$
    d. $(x^3 + x^2 + x + 1) / (x^2 + 1)$
25. Answer the following questions:
    a. What is the polynomial representation of 10111O?
    b. What is the result of shifting 101110 three bits to the left?
    c. Repeat part b using polynomials.
    d. What is the result of shifting 101110 four bits to the right?
    e. Repeat part d using polynomials.
26. Which of the following CRC generators guarantee the detection of a single bit error?
    a. $x^3 + x + 1$
    b. $x^4 + xl$
    c. 1
    d. $x^2 + 1$
27. Referring to the CRC-8 polynomial in Table 10.7, answerthe following questions:
    a. Does it detect a single error? Defend your answer.
    b. Does it detect a burst error of size 6? Defend your answer.
    c. What is the probability of detecting a burst error of size 9?
    d. What is the probability of detecting a burst error of size 15?
28. Referring to the CRC-32 polynomial in Table 10.7, answer the following questions:
    a. Does it detect a single error? Defend your answer.
    b. Does it detect a burst error of size 16? Defend your answer.
    c. What is the probability of detecting a burst error of size 33?
    d. What is the probability of detecting a burst error of size 55?

29. Assuming even parity, find the parity bit for each of the following data units.
    a. 1001011
    b. 0001100
    c. 1000000
    d. 1110111
30. Given the dataword 1010011110 and the divisor 10111,
    a. Show the generation of the codeword at the sender site (using binary division).
    b. Show the checking of the codeword at the receiver site (assume no error).
31. Repeat Exercise 30 using polynomials.
32. A sender needs to send the four data items Ox3456, OxABCC, Ox02BC, and OxEEEE. Answer the following:
    a. Find the checksum at the sender site.
    b. Find the checksum at the receiver site if there is no error.
    c. Find the checksum at the receiver site if the second data item is changed to OxABCE.
    d. Find the checksum at the receiver site if the second data item is changed to OxABCE and the third data item is changed to Ox02BA.
33. This problem shows a special case in checksum handling. A sender has two data items to send: Ox4567 and OxBA98. What is the value of the checksum?

CHAPTER 11

Data Link Control

The two main functions of the data link layer are data link control and media access control. The first, data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication. We discuss this functionality in this chapter. The second function of the data link layer is media access control, or how to share the link. We discuss this functionality in Chapter 12.

Data link control functions include framing, flow and error control, and software-implemented protocols that provide smooth and reliable transmission of frames between nodes. In this chapter, we first discuss framing, or how to organize the bits that are carried by the physical layer. We then discuss flow and error control. A subset of this topic, techniques for error detection and correction, was discussed in Chapter 10.

To implement data link control, we need protocols. Each protocol is a set of rules that need to be implemented in software and run by the two nodes involved in data exchange at the data link layer. We discuss five protocols: two for noiseless (ideal) channels and three for noisy (real) channels. Those in the first category are not actually implemented, but provide a foundation for understanding the protocols in the second category.

After discussing the five protocol designs, we show how a bit-oriented protocol is actually implemented by using the High-level Data Link Control (HDLC) Protocol as an example. We also discuss a popular byte-oriented protocol, Point-to-Point Protocol (PPP).

11.1 FRAMING

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination. The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.

The data link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses since the postal system is a many-to-many carrier facility.

Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Fixed-Size Framing

Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells. We discuss ATM in Chapter 18.

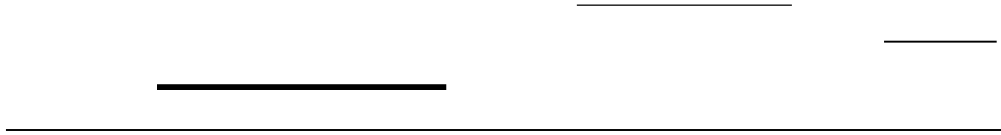
Variable-Size Framing

Our main discussion in this chapter concerns variable-size framing, prevalent in local-area networks. In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

Character-Oriented Protocols

In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII (see Appendix A). The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (I-byte) flag is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure 11.1 shows the format of a frame in a character-oriented protocol.

Figure 11.1 *A frame in a character-oriented protocol*

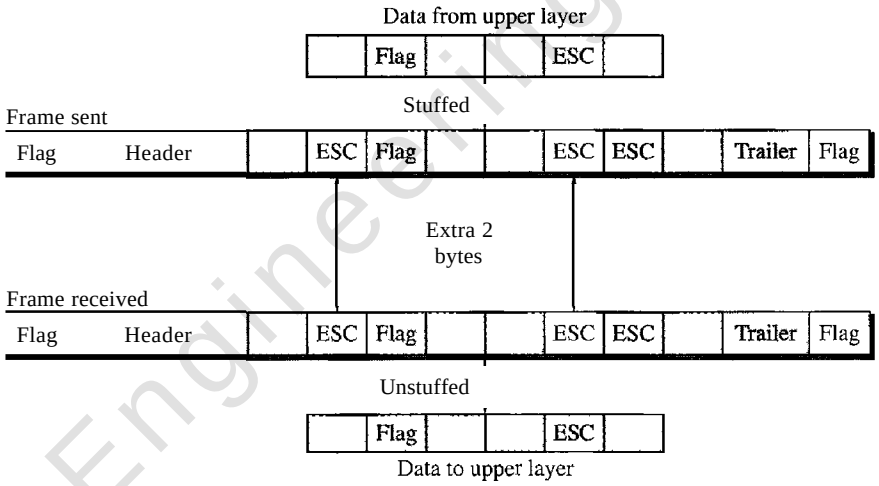


Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to

character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a flag? The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text. Figure 11.2 shows the situation.

Figure 11.2 *Byte stuffing and unstuffing*



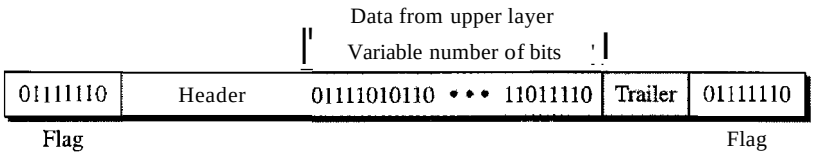
> Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that in general, the tendency is moving toward the bit-oriented protocols that we discuss next.

Bit-Oriented Protocols

In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

Figure 11.3 *A frame in a bit-oriented protocol*

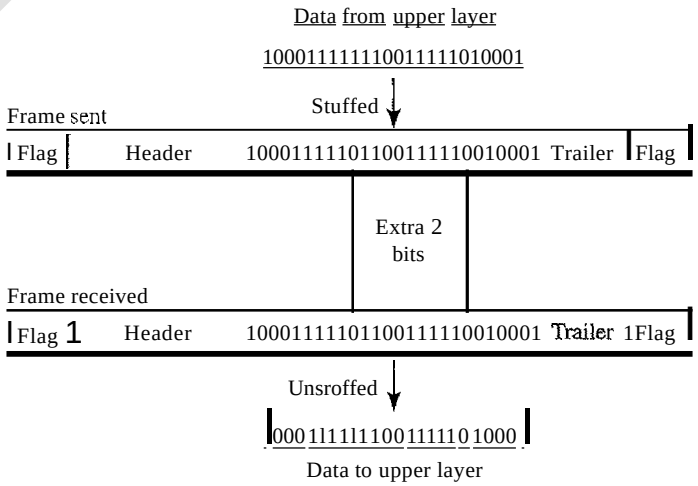


This flag can create the same type of problem we saw in the byte-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of I byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing. In bit stuffing, if a 0 and five consecutive I bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 18 follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.

Figure 11.4 *Bit stuffing and unstuffing*



This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

11.2 FLOW AND ERROR CONTROL

Data communication requires at least two devices working together, one to send and the other to receive. Even such a basic arrangement requires a great deal of coordination for an intelligible exchange to occur. The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.

Flow Control

Flow control coordinates the amount of data that can be sent before receiving an acknowledgment and is one of the most important duties of the data link layer. In most protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. The flow of data must not be allowed to overwhelm the receiver. Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily. Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called a *buffer,* reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

Error Control

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term *error control* refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

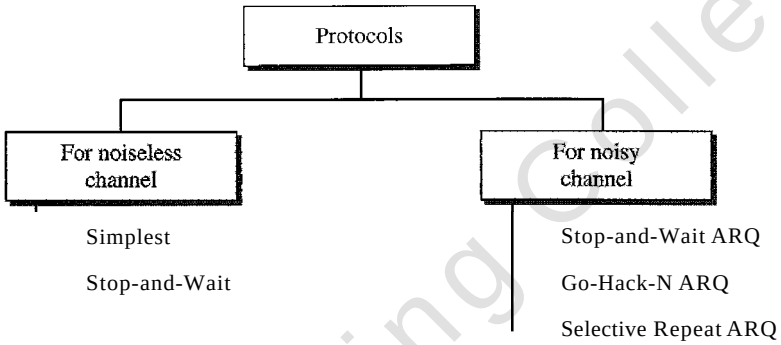
11.3 PROTOCOLS

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our

discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

We divide the discussion of protocols into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels. The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels. Figure 11.5 shows the classifications.

Figure 11.5 *Taxonomy ofprotocols discussed in this chapter*



There is a difference between the protocols we discuss here and those used in real networks. All the protocols we discuss are unidirectional in the sense that the data frames travel from one node, called the sender, to another node, called the receiver. Although special frames, called acknowledgment (ACK) and negative acknowledgment (NAK) can flow in the opposite direction for flow and error control purposes, data flow in only one direction.

In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions. In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called piggybacking. Because bidirectional protocols are more complex than unidirectional ones, we chose the latter for our discussion. If they are understood, they can be extended to bidirectional protocols. We leave this extension as an exercise.

11.4 NOISELESS CHANNELS

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel. The first is a protocol that does not use flow control; the second is the one that does. Of course, neither has error control because we have assumed that the channel is a perfect noiseless channel.

Simplest Protocol

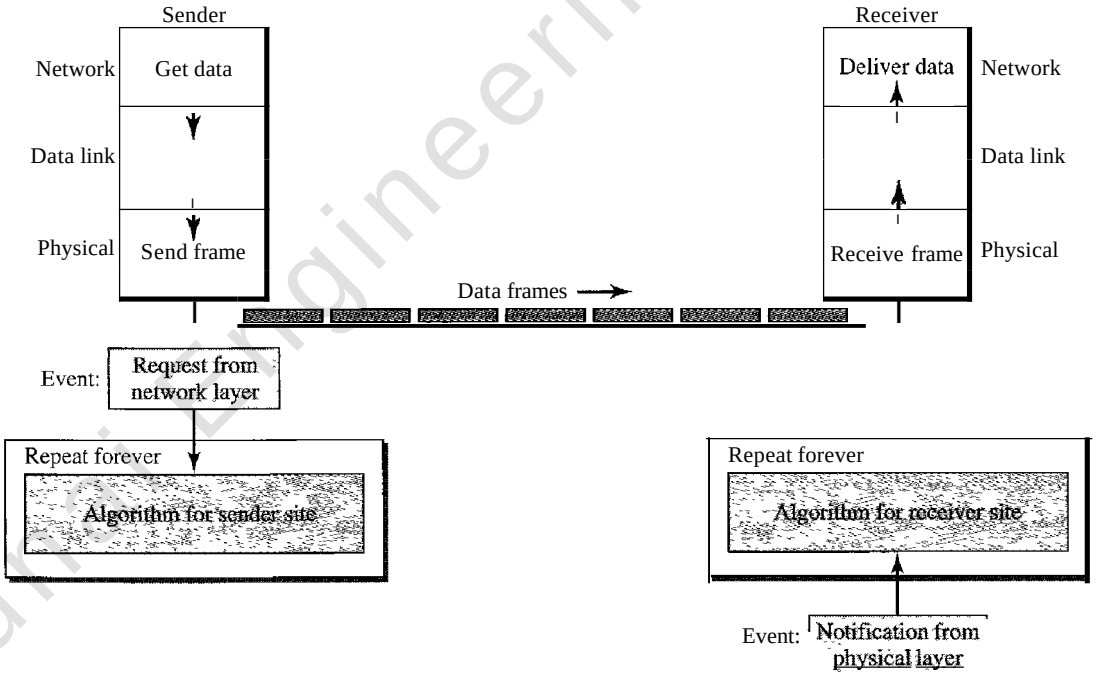
Our first protocol, which we call the Simplest Protocol for lack of any other name, is one that has no flow or en'or control. Like other protocols we will discuss in this chapter, it is a unidirectional protocol in which data frames are traveling in only one direction-from the

sender to receiver. We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.

Design

There is no need for flow control in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer. The data link layers of the sender and receiver provide transmission services for their network layers. The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits. Figure 11.6 shows a design.

Figure 11.6 *The design of the simplest protocol with no flow or error control*



We need to elaborate on the procedure used by both data link layers. The sender site cannot send a frame until its network layer has a data packet to send. The receiver site cannot deliver a data packet to its network layer until a frame arrives. If the protocol is implemented as a procedure, we need to introduce the idea of events in the protocol. The procedure at the sender site is constantly running; there is no action until there is a request from the network layer. The procedure at the receiver site is also constantly rulming, but there is no action until notification from the physical layer arrives. Both procedures are constantly running because they do not know when the corresponding events will occur.

Algorithms

Algorithm 11.1 shows the procedure at the sender site.

Algorithm 11.1 *Sender-site algorithm for the simplest protocol*

```
1  while (true)                          // Repeat forever
2  {
3    WaitForEvent()i                     // Sleep until an event occurs
4    if(Event(RequestToSend»             //There is a packet to send
5    {
6        GetData()i
7        MakeFrame()i
8        SendFrame()i                    //Send the frame
9    }
10 }
```

Analysis The algorithm has an infinite loop, which means lines 3 to 9 are repeated forever once the program starts. The algorithm is an event-driven one, which means that it *sleeps* (line 3) until an event *wakes* it *up* (line 4). This means that there may be an undefined span of time between the execution of line 3 and line 4; there is a gap between these actions. When the event, a request from the network layer, occurs, lines 6 though 8 are executed. The program then repeats the loop and again sleeps at line 3 until the next occurrence of the event. We have written pseudocode for the main process. We do not show any details for the modules GetData, MakeFrame, and SendFrame. GetDataO takes a data packet from the network layer, MakeFrameO adds a header and delimiter flags to the data packet to make a frame, and SendFrameO delivers the frame to the physical layer for transmission.

Algorithm 11.2 shows the procedure at the receiver site.

Algorithm 11.2 *Receiver-site algorithm for the simplest protocol*

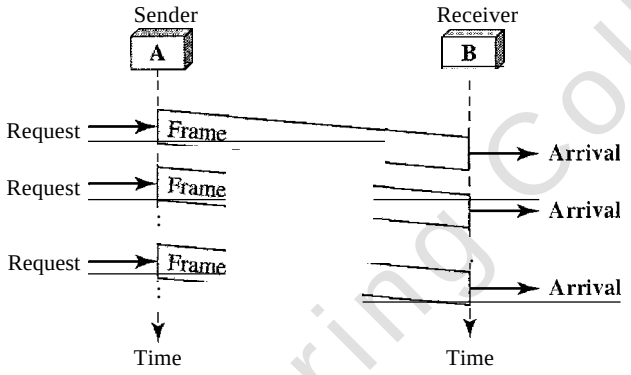
```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent()i                    II Sleep until an event occurs
4    if(Event(ArrivalNotification»      IIData frame arrived
5    {
6        ReceiveFrame()i
7        ExtractData()i
8        DeliverData ()i                //Deliver data to network layez
9    }
10 }
```

Analysis This algorithm has the same format as Algorithm 11.1, except that the direction of the frames and data is upward. The event here is the arrival of a data frame. After the event occurs, the data link layer receives the frame from the physical layer using the ReceiveFrameO process, extracts the data from the frame using the ExtractDataO process, and delivers the data to the network layer using the DeliverDataO process. Here, we also have an event-driven algorithm because the algorithm never knows when the data frame will arrive.

Example 11.1

Figure 11.7 shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.

Figure 11.7 *Flow diagram for Example 11.1*



Stop-and-Wait Protocol

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames,we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

The protocol we discuss now is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

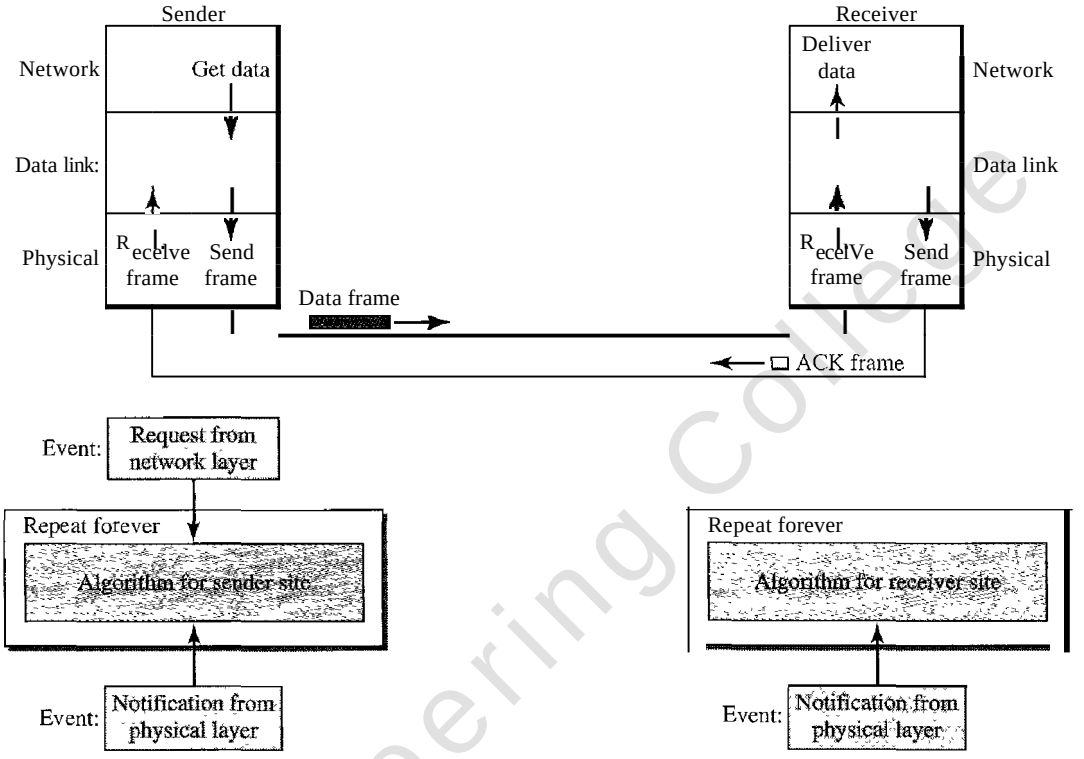
Design

Figure 11.8 illustrates the mechanism. Comparing this figure with Figure 11.6, we can see the traffic on the forward channel (from sender to receiver) and the reverse channel. At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. We therefore need a half-duplex link.

Algorithms

Algorithm 11.3 is for the sender site.

Figure 11.8 *Design of Stop-and-Wait Protocol*



Algorithm 11.3 *Sender-site algorithm for Stop-and-Wait Protocol*

```
1  while(true)                    IIRepeat forever
2  canSend = true                 IIAllow the first frame to go
3  {
4    WaitForEvent()i              II Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend}
6    {
7       GetData()i
8       MakeFrame();
9       SendFrame()i              I/Send the data frame
10      canSend = false;          I/cannot send until ACK arrives
11   }
12   WaitForEvent()i              II Sleep until an event occurs
13   if(Event(ArrivalNotification) /1 An ACK has arrived
14    {
15      ReceiveFrame();           I/Receive the ACK frame
16      canSend = true;
17    }
18 }
```

Analysis Here two events can occur: a request from the network layer or an arrival notification from the physical layer. The responses to these events must alternate. In other words, after a frame is sent, the algorithm must ignore another network layer request until that frame is

acknowledged. We know that two arrival events cannot happen one after another because the channel is error-free and does not duplicate the frames. The requests from the network layer, however, may happen one after another without an arrival event in between. We need somehow to prevent the immediate sending of the data frame. Although there are several methods, we have used a simple *canSend* variable that can either be true or false. When a frame is sent, the variable is set to false to indicate that a new network request cannot be sent until *canSend* is true. When an ACK is received, canSend is set to true to allow the sending of the next frame.

Algorithm 11.4 shows the procedure at the receiver site.

Algorithm 11.4 *Receiver-site algorithm for Stop-and-Wait Protocol*

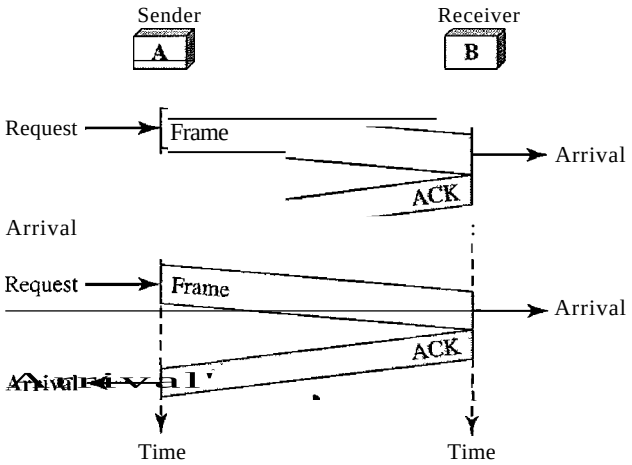
```
1  while (true)                          IIRepeat forever
2  {
3    WaitForEvent();                     II Sleep until an event occurs
4    if(Event(ArrivalNotification)}      IIData frame arrives
5    {
6        ReceiveFrame(};
7        ExtractData(}i
8        Deliver(data};                  /IDeliver data to network layex
9        SendFrame();                    IISend an ACK frame
10   }
11 }
```

Analysis This is very similar to Algorithm 11.2 with one exception. After the data frame arrives, the receiver sends an ACK frame (line 9) to acknowledge the receipt and allow the sender to send the next frame.

Example 11.2

Figure 11.9 shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.

Figure 11.9 *Flow diagram for Example 11.2*



11.5 NOISY CHANNELS

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and-Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

To detect and correct corrupted frames, we need to add redundancy bits to our data frame (see Chapter 10). When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.

The comlpted and lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. The ACK frame for this protocol has a sequence number field. In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

Sequence Numbers

As we discussed, the protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.

One important consideration is the range of the sequence numbers. Since we want to minimize the frame size, we look for the smallest range that provides unambiguous

communication. The sequence numbers of course can wrap around. For example, if we decide that the field is *m* bits long, the sequence numbers start from 0, go to $2^m - 1$, and then are repeated.

Let us reason out the range of sequence numbers we need. Assume we have used *x* as a sequence number; we only need to use $x + 1$ after that. There is no need for $x + 2$. To show this, assume that the sender has sent the frame numbered *x*. Three things can happen.

1. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment. The acknowledgment arrives at the sender site, causing the sender to send the next frame numbered $x + 1$.
2. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost. The sender resends the frame (numbered *x)* after the time-out. Note that the frame here is a duplicate. The receiver can recognize this fact because it expects frame $x + I$ but frame *x* was received.
3. The frame is corrupted or never arrives at the receiver site; the sender resends the frame (numbered *x)* after the time-out.

We can see that there is a need for sequence numbers *x* and $x + I$ because the receiver needs to distinguish between case 1 and case 2. But there is no need for a frame to be numbered $x + 2$. In case 1, the frame can be numbered *x* again because frames *x* and $x + 1$ are acknowledged and there is no ambiguity at either site. In cases 2 and 3, the new frame is $x + I$, not $x + 2$. If only *x* and $x + 1$ are needed, we can let $x = 0$ and $x + I = 1$. This means that the sequence is 0, I, 0, I, 0, and so on. Is this pattern familiar? This is modulo-2 arithmetic as we saw in Chapter 10.

In Stop-and-Wait **ARQ,** we use sequence numbers to number the frames. The sequence numbers are based on modul0-2 arithmetic.

Acknowledgment Numbers

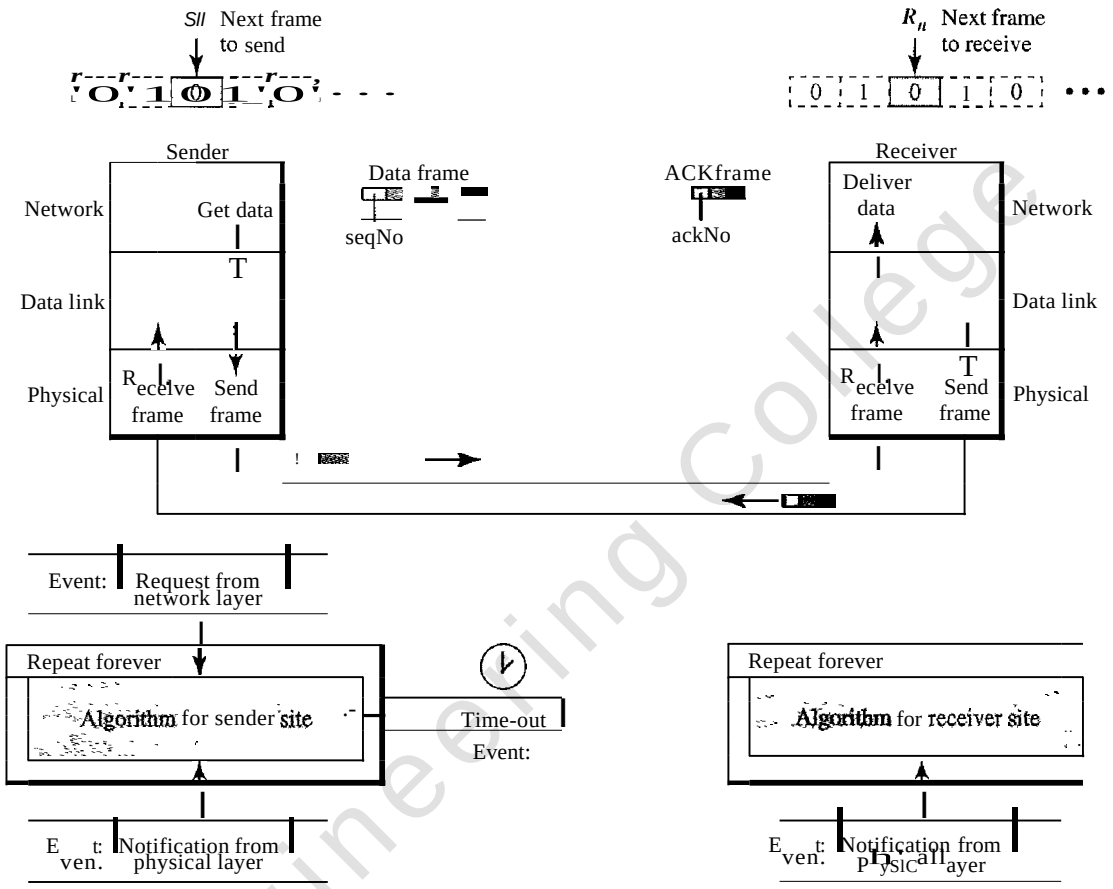
Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

In **Stop-and-Wait ARQ,** the acknowledgment number always announces in modul0-2 arithmetic the sequence number of the next frame expected.

Design

Figure 11.10 shows the design of the Stop-and-Wait ARQ Protocol. The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number). The sender has a control variable, which we call *Sn* (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).

Figure 11.10 *Design of the Stop-and-Wait ARQ Protocol*



The receiver has a control variable, which we call R_n (receiver, next frame expected), that holds the number of the next frame expected. When a frame is sent, the value of *Sn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of R_n is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable *Sn* points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; R_n points to the slot that matches the sequence number of the expected frame.

Algorithms

Algorithm 11.5 is for the sender site.

Algorithm 11.5 *Sender-site algorithm for Stop-and-Wait ARQ*

```
1   n = 0;                          // Frame 0 should be sent first
2   anSend = true;                  // Allow the first request to go
3   hile(true)                      // Repeat forever
4   {
5     WaitForEvent();               // Sleep until an event occurs
```

Algorithm 11.5 *Sender-site algorithm for Stop-and-Wait ARQ (continued)*

```
 6    if (Event (RequestToSend) AND canSend)
 7    {
 8       GetData () i
 9       MakeFrame (Sn) ;                    //The seqNo is Sn
10       StoreFrame(Sn);                     //Keep copy
11       SendFrame(Sn) ;
12       StartTimerO;
13       Sn = Sn + 1;
14       canSend = false;
15    }
16    WaitForEvent();                        II Sleep
17      if (Event (ArrivaINotification)      II An ACK has arrived
18      {
19        ReceiveFrame(ackNo);               //Receive the ACE fram
20        if(not corrupted AND ackNo     Sn) //Valid ACK
21          {
22            Stoptimer{};
23            PurgeFrame(Sn_l);              //Copy is not needed
24            canSend = true;
25          }
26       }
27
28      if (Event (TimeOUt)                  II The timer expired
29      {
30       StartTimer();
31       ResendFrame(Sn_l);                 //Resend a copy check
32      }
33 }
```

Analysis We first notice the presence of *Sn'* the sequence number of the next frame to be sent. This variable is initialized once (line 1), but it is incremented every time a frame is sent (line 13) in preparation for the next frame. However, since this is modulo-2 arithmetic, the sequence numbers are 0, 1,0, 1, and so on. Note that the processes in the first event (SendFrame, StoreFrame, and PurgeFrame) use an *Sn* defining the frame sent out. We need at least one buffer to hold this frame until we are sure that it is received safe and sound. Line 10 shows that before the frame is sent, it is stored. The copy is used for resending a corrupt or lost frame. We are still using the canSend variable to prevent the network layer from making a request before the previous frame is received safe and sound. If the frame is not corrupted and the ackNo of the ACK frame matches the sequence number of the next frame to send, we stop the timer and purge the copy of the data frame we saved. Otherwise, we just ignore this event and wait for the next event to happen. After each frame is sent, a timer is started. When the timer expires (line 28), the frame is resent and the timer is restarted.

Algorithm 11.6 shows the procedure at the receiver site.

Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol*

```
1     = 0;                          II Frame 0 expected to arrive firs
2   hile (true)
3  {
4    WaitForEvent();                II Sleep until an event occurs
```

Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol (continued)*

```
5   if(Event(Arriva1Notification»     //Data frame arrives
6   {
7       ReceiveFrame()i
8       if(corrupted(frame»i
9          sleep()i
10      if(seqNo == Rn)               //Valid data frame
11      {
12       ExtractData();
13        De1iverData()i              //Deliver data
14        Rn = Rn + 1;
15      }
16       SendFrame(Rn);               //Send an ACK
17  }
18 }
```

Analysis This is noticeably different from Algorithm 11.4. First, all arrived data frames that are corrupted are ignored. If the seqNo of the frame is the one that is expected (R_n), the frame is accepted, the data are delivered to the network layer, and the value of R_n is incremented. However, there is one subtle point here. Even if the sequence number of the data frame does not match the next frame expected, an ACK is sent to the sender. This ACK, however, just reconfirms the previous ACK instead of confirming the frame received. This is done because the receiver assumes that the previous ACK might have been lost; the receiver is sending a duplicate frame. The resent ACK may solve the problem before the time-out does it.

Example 11.3

Figure 11.11 shows an example of Stop-and-Wait ARQ. Frame a is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame a is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Efficiency

The Stop-and-WaitARQ discussed in the previous section is very inefficient if our channel is *thick* and *long.* By *thick,* we mean that our channel has a large bandwidth; by *long,* we mean the round-trip delay is long. The product of these two is called the bandwidth-delay product, as we discussed in Chapter 3. We can think of the channel as a pipe. The bandwidth-delay product then is the volume of the pipe in bits. The pipe is always there. If we do not use it, we are inefficient. The bandwidth-delay product is a measure of the number of bits we can send out of our system while waiting for news from the receiver.

Example 11.4

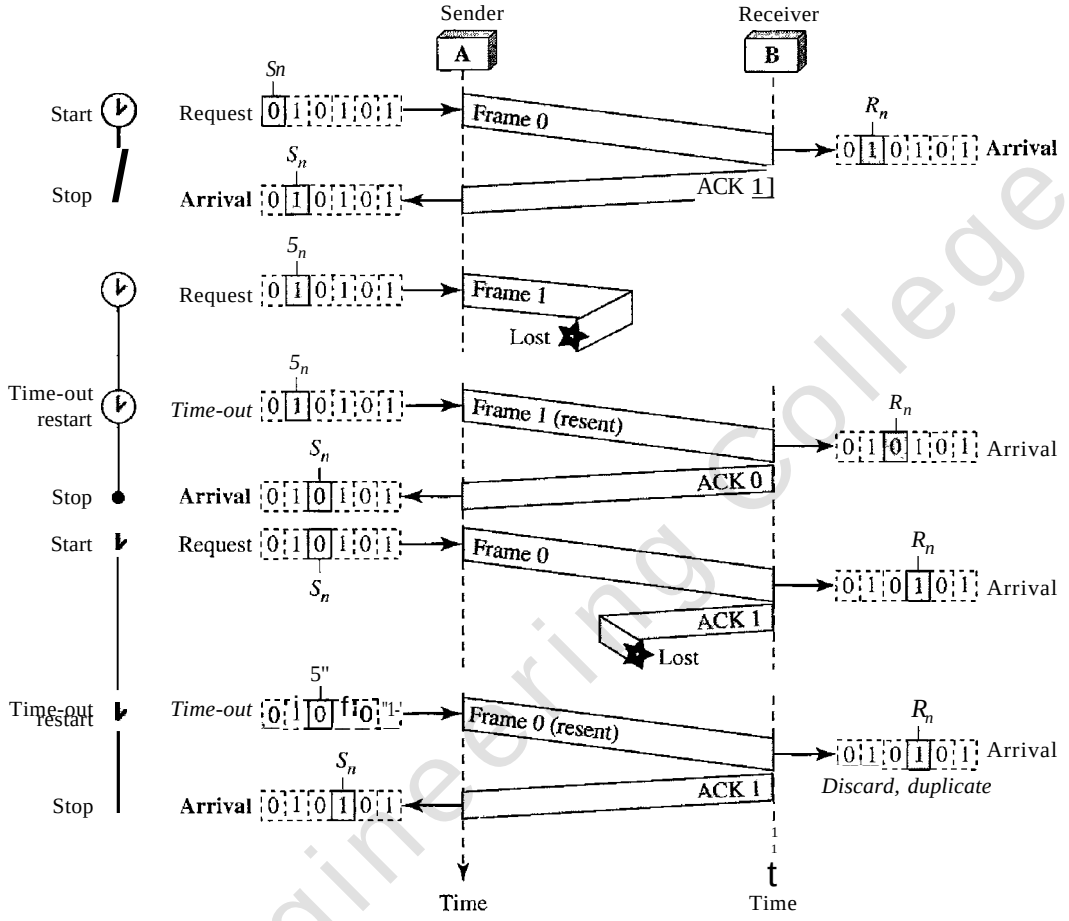
Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

Solution

The bandwidth-delay product is

$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}$$

Figure 11.11 *Flow diagram for Example 11.3*



The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again. However, the system sends only 1000 bits. We can say that the link utilization is only 1000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link.

Example 11.5

What is the utilization percentage of the link in Example 11.4 if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

The bandwidth-delay product is still 20,000 bits. The system can send up to 15 frames or 15,000 bits during a round trip. This means the utilization is 15,000/20,000, or 75 percent. Of course, if there are damaged frames, the utilization percentage is much less because frames have to be resent.

Pipelining

In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining. There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent. However, pipelining does apply to our next two protocols because

several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second.

The first is called Go-Back-N Automatic Repeat Request (the rationale for the name will become clear later). In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to $2^m - 1$. For example, if m is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

In other words, the sequence numbers are modulo-2^m.

In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

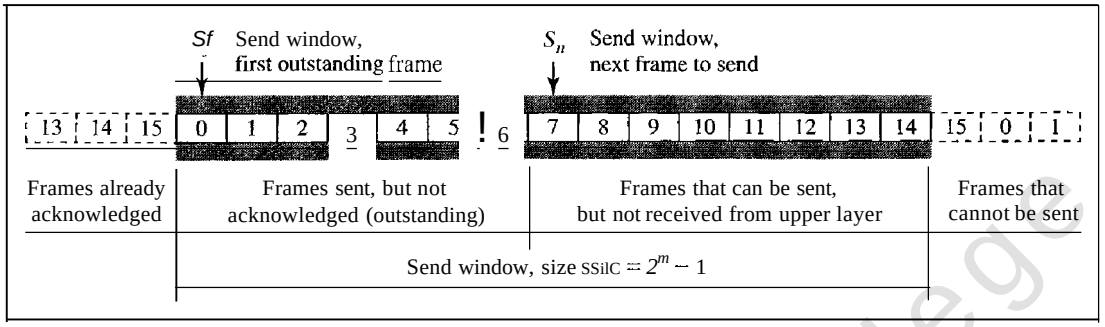
Sliding Window

In this protocol (and the next), the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window. We discuss both here.

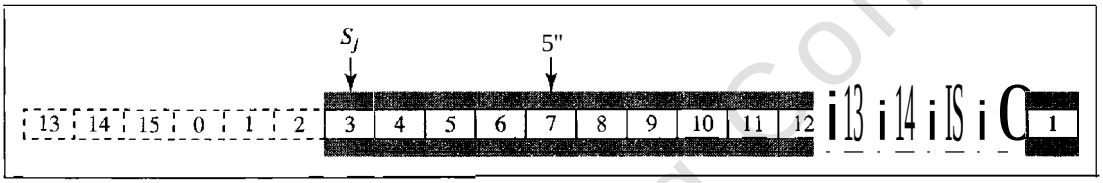
The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is $2^m - 1$ for reasons that we discuss later. In this chapter, we let the size be fixed and set to the maximum value, but we will see in future chapters that some protocols may have a variable window size. Figure 11.12 shows a sliding window of size 15 ($m = 4$).

The window at any time divides the possible sequence numbers into four regions. The first region, from the far left to the left wall of the window, defines the sequence

Figure 11.12 *Send window for Go-Back-NARQ*



a. Send window before sliding



b. Send window after sliding

numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region, colored in Figure 11.12a, defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

The window itself is an abstraction; three variables define its size and location at any time. We call these variables *Sf(send* window, the first outstanding frame), *Sn* (send window, the next frame to be sent), and Ssize (send window, size). The variable *Sf* defines the sequence number of the first (oldest) outstanding frame. The variable *Sn* holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window, which is fixed in our protocol.

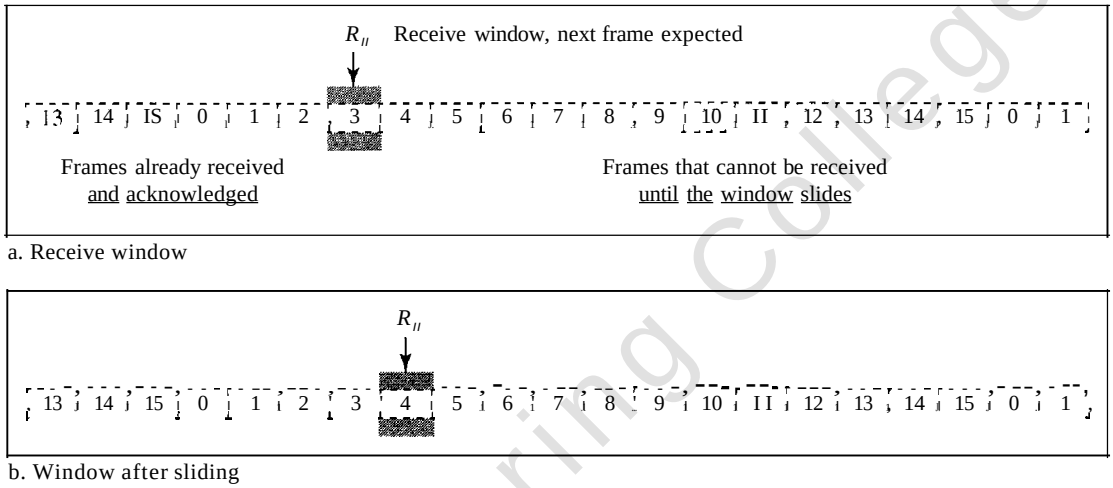
The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f, S_n, and *Ssize'*

Figure 11.12b shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. As we will see shortly, the acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure 11.12b, frames 0, I, and 2 are acknowledged, so the window has slid to the right three slots. Note that the value of *Sf* is 3 because frame 3 is now the first outstanding frame.

The send window can slide one or more slots when a valid acknowledgment arrives.

The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent. The size of the receive window is always I. The receiver is always looking for the arrival of a specific frame. Any frame arriving out of order is discarded and needs to be resent. Figure 11.13 shows the receive window.

Figure 11.13 *Receive window for Go-Back-NARQ*



The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

Note that we need only one variable R_n (receive window, next frame expected) to define this abstraction. The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of R_n is accepted and acknowledged.

The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.

Timers

Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

Acknowledgment

The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of

the receiver causes the timer of the unacknowledged frame at the sender site to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

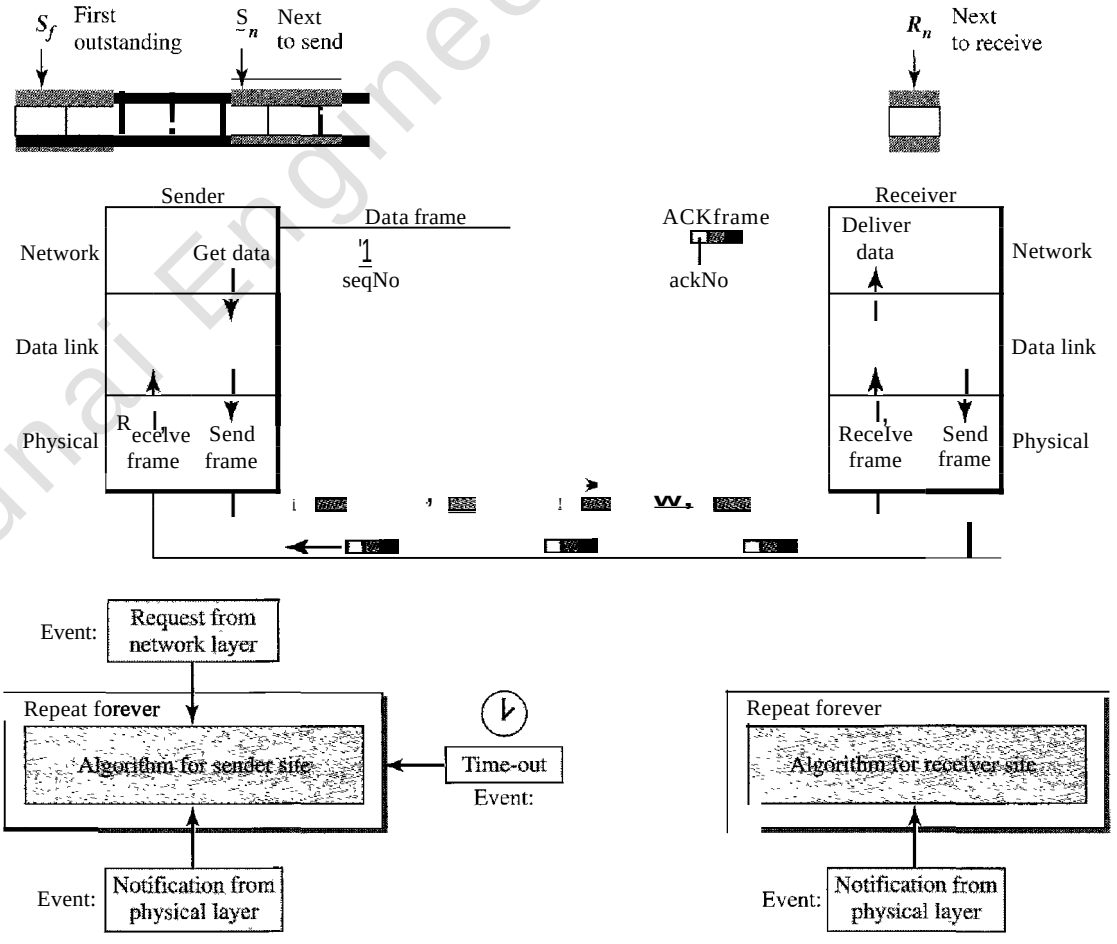
Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3, 4, 5, and 6 again. That is why the protocol is called *Go-Back-N* ARQ.

Design

Figure 11.14 shows the design for this protocol. As we can see, multiple frames can be in transit in the forward direction, and mUltiple acknowledgments in the reverse direction. The idea is similar to Stop-and-Wait ARQ; the difference is that the send

Figure **11.14** *Design of Go-Back-N ARQ*

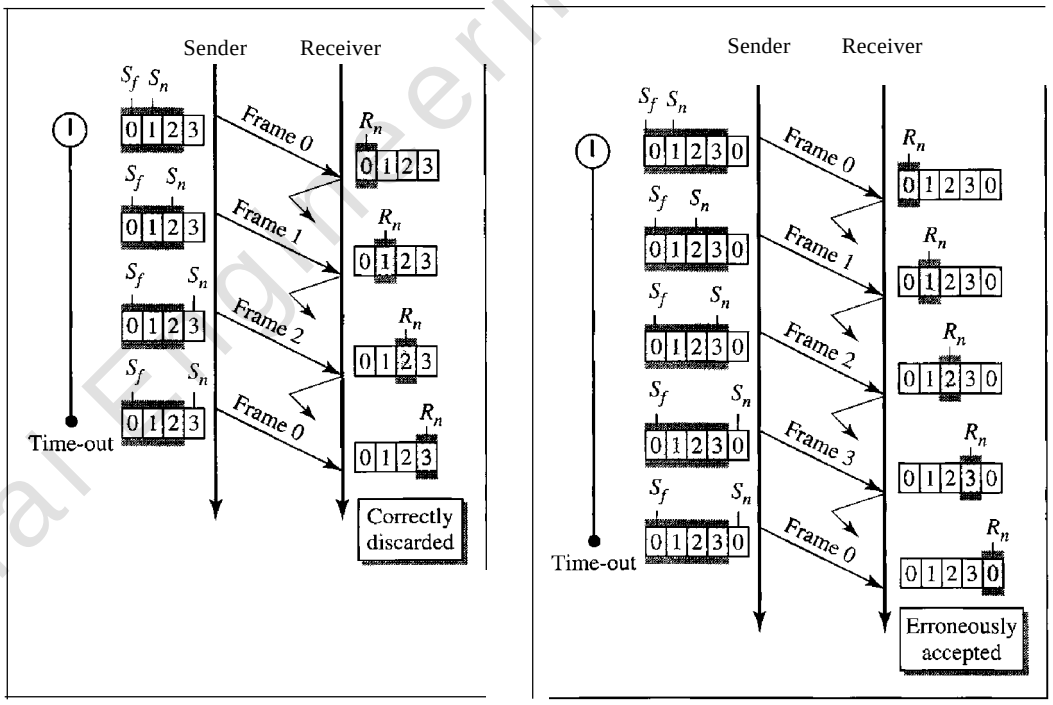


window allows us to have as many frames in transition as there are slots in the send window.

Send Window Size

We can now show why the size of the send window must be less than 2^m. As an example, we choose $m = 2$, which means the size of the window can be $2^m - 1$, or 3. Figure 11.15 compares a window size of 3 against a window size of 4. If the size of the window is 3 (less than 2^2) and all three acknowledgments are lost, the frame timer expires and all three frames are resent. The receiver is now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded. On the other hand, if the size of the window is 4 (equal to 2^2) and all acknowledgments are lost, the sender will send a duplicate of frame 0. However, this time the window of the receiver expects to receive frame 0, so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is an error.

Figure 11.15 *Window size for Go-Back-N ARQ*



a. Window size < 2^m

b. Window size = 2^m

In Go-Back-N ARQ, the size of the send window must be less than 2^m; the size of the receiver window is always 1.

Algorithms

Algorithm 11.7 shows the procedure for the sender in this protocol.

Algorithm **11.7** *Go-Back-N sender algorithm*

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend»             //A packet to send
9    {
10       if(Sn-Sf >= Sw)                 IIIf window is full
11             Sleep();
12       GetData();
13       MakeFrame(Sn);
14       StoreFrame(Sn);
15       SendFrame(Sn);
16       Sn = Sn + 1;
17       if(timer not running)
18            StartTimer{);
19   }
20
21   if{Event{ArrivalNotification»       IIACK arrives
22   {
23       Receive(ACK);
24       if{corrupted{ACK»
25            Sleep();
26       if{{ackNo>sf)&&{ackNO<=Sn»      IIIf a valid ACK
27       While(Sf <= ackNo)
28        {
29         PurgeFrame{Sf);
30         Sf = Sf + 1;
31        }
32        StopTimer();
33   }
34
35   if{Event{TimeOut»                   liThe timer expires
36   {
37    StartTimer();
38    Temp = Sf;
39    while{Temp < Sn);
40      {
41       SendFrame(Sf);
42       Sf = Sf + 1;
43      }
44   }
45 }
```

Analysis This algorithm first initializes three variables. Unlike Stop-and-Wait ARQ, this protocol allows several requests from the network layer without the need for other events to occur; we just need to be sure that the window is not full (line 12). In our approach, if the window is full,

the request is just ignored and the network layer needs to try again. Some implementations use other methods such as enabling or disabling the network layer. The handling of the arrival event is more complex than in the previous protocol. If we receive a corrupted ACK, we ignore it. If the adeNa belongs to one of the outstanding frames, we use a loop to purge the buffers and move the left wall to the right. The time-out event is also more complex. We first start a new timer. We then resend all outstanding frames.

Algorithm 11.8 is the procedure at the receiver site.

Algorithm 11.8 *Go-Back-N receiver algorithm*

```
1  Rn = 0;
2
3  while (true)                          IIRepeat forever
4  {
5    WaitForEvent();
6
7    if(Event{ArrivalNotification»       /Data frame arrives
8    (
9       Receive(Frame);
10      if(corrupted(Frame»
11           Sleep(};
12      if(seqNo == Rn)                  IIIf expected frame
13      {
14        DeliverData()i                 IIDeliver data
15        Rn = Rn + 1;                   IISlide window
16        SendACK(Rn);
17      }
18   }
19 }
```

Analysis This algorithm is simple. We ignore a corrupt or out-of-order frame. If a frame arrives with an expected sequence number, we deliver the data, update the value of R_n, and send an ACK with the ackNa showing the next frame expected.

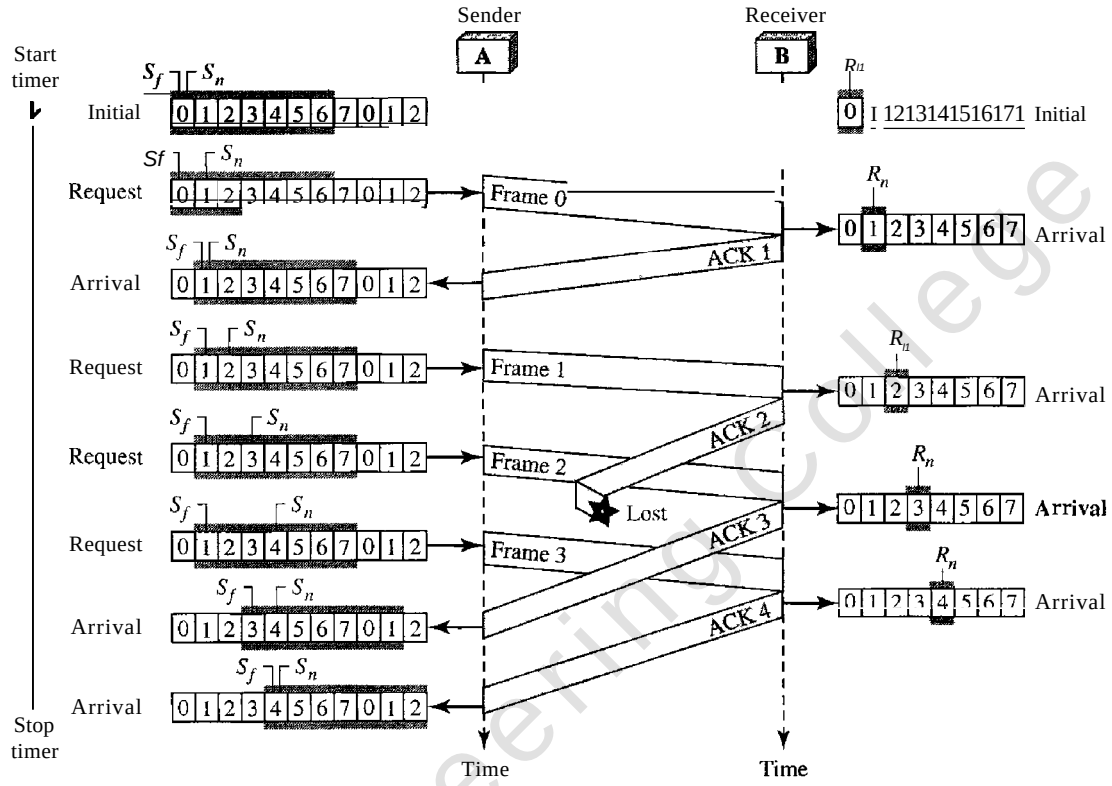
Example 11.6

Figure 11.16 shows an example of Go-Back-*N*. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.

After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK3.

There are four receiver events, all triggered by the arrival of frames from the physical layer.

Figure 11.16 *Flow diagram for Example 11.6*



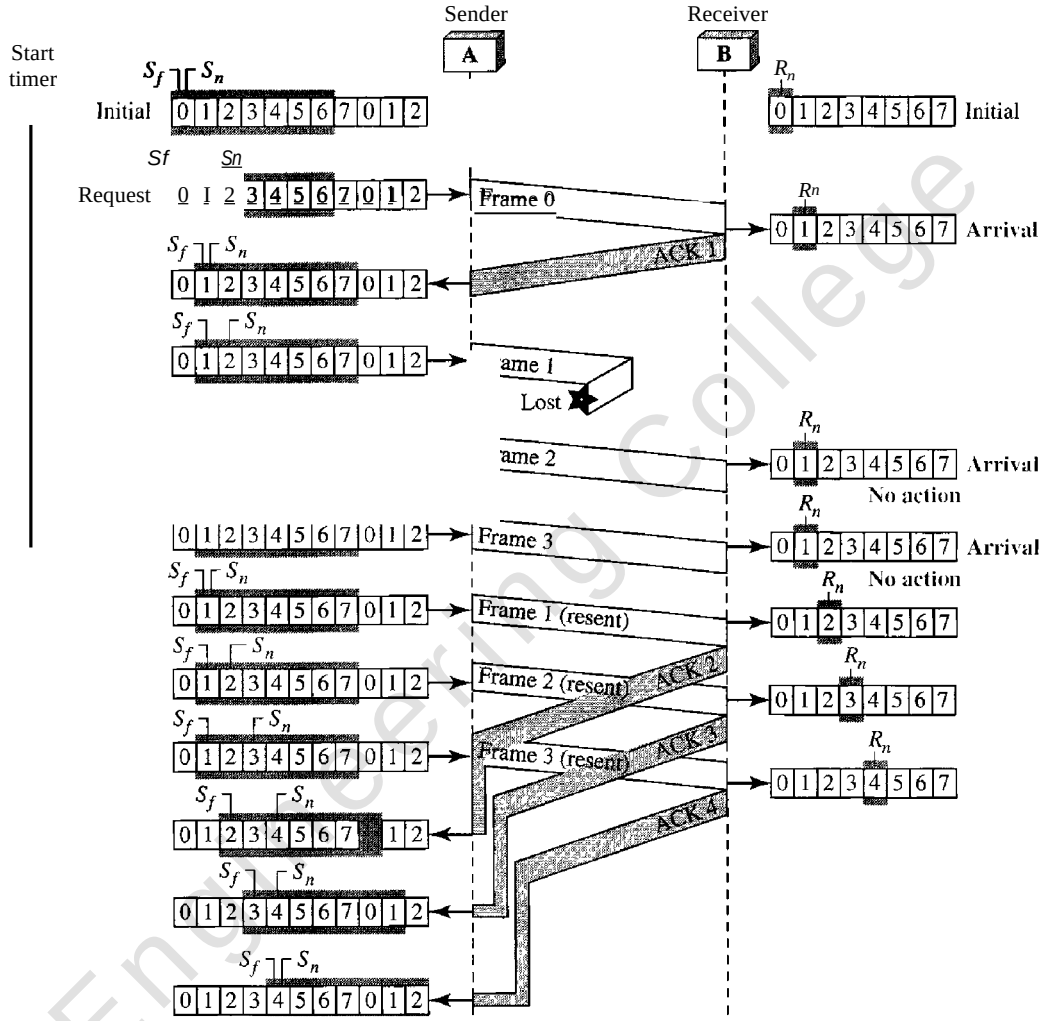
Example 11.7

Figure 11.17 shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order (frame 1 is expected). The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames l, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with sending frame 3. The physica1layer must wait until this event is completed and the data link layer goes back to its sleeping state. We have shown a vertical line to indicate the delay. It is the same story with ACK 3; but when ACK 3 arrives, the sender is busy responding to ACK 2. It happens again when ACK 4 arrives. Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.

Go-Back-N ARQ Versus Stop-and-Wait ARQ

The reader may find that there is a similarity between *Go-Back-N* ARQ and Stop-and-Wait ARQ. We can say that the Stop-and-WaitARQ Protocol is actually a *Go-Back-N* ARQ in which there are only two sequence numbers and the send window size is 1. In other words, $m = 1$, $2^m - 1 = 1$. In *Go-Back-N* ARQ, we said that the addition is modulo-2^m; in Stop-and-Wait ARQ it is 2, which is the same as 2^m when $m = 1$.

Figure 11.17 *Flow diagram for Example 11.7*



Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

Selective Repeat Automatic Repeat Request

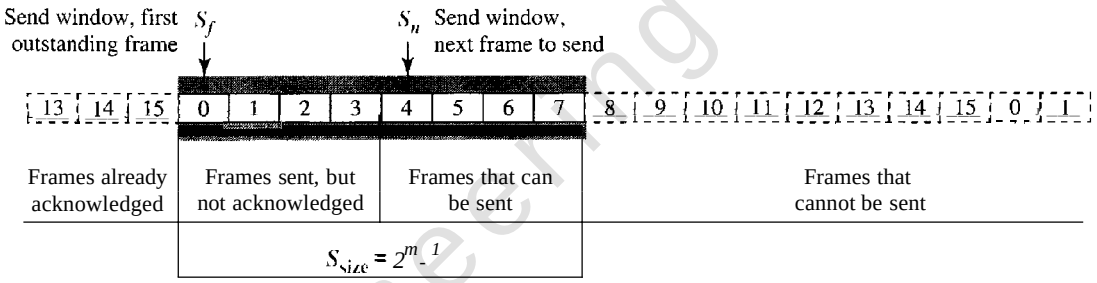
Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ. It is more efficient for noisy links, but the processing at the receiver is more complex.

Windows

The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are differences between the windows in this protocol and the ones in Go-Back-N. First, the size of the send window is much smaller; it is 2^{m-1}. The reason for this will be discussed later. Second, the receive window is the same size as the send window.

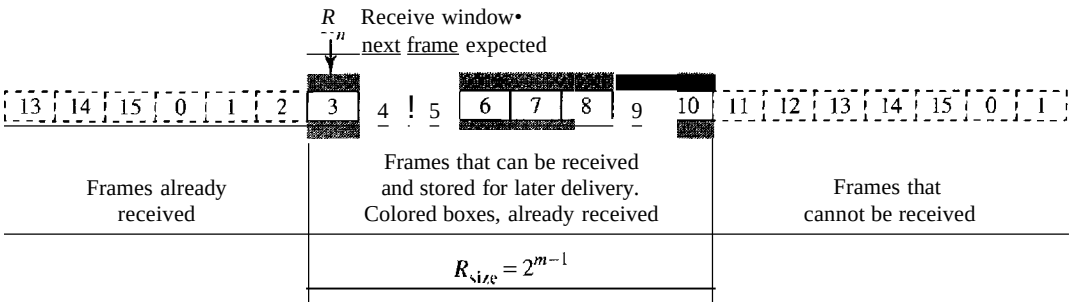
The send window maximum size can be 2^{m-1}. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the *Go-Back-N* Protocol). The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this. The protocol uses the same variables as we discussed for Go-Back-N. We show the Selective Repeat send window in Figure 11.18 to emphasize the size. Compare it with Figure 11.12.

Figure 11.18 *Send window for Selective Repeat ARQ*



The receive window in Selective Repeat is totally different from the one in Go-Back-N. First, the size of the receive window is the same as the size of the send window (2^{m-1}). The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. We need, however, to mention that the receiver never delivers packets out of order to the network layer. Figure 11.19 shows the receive window in this

Figure 11.19 *Receive window for Selective Repeat ARQ*

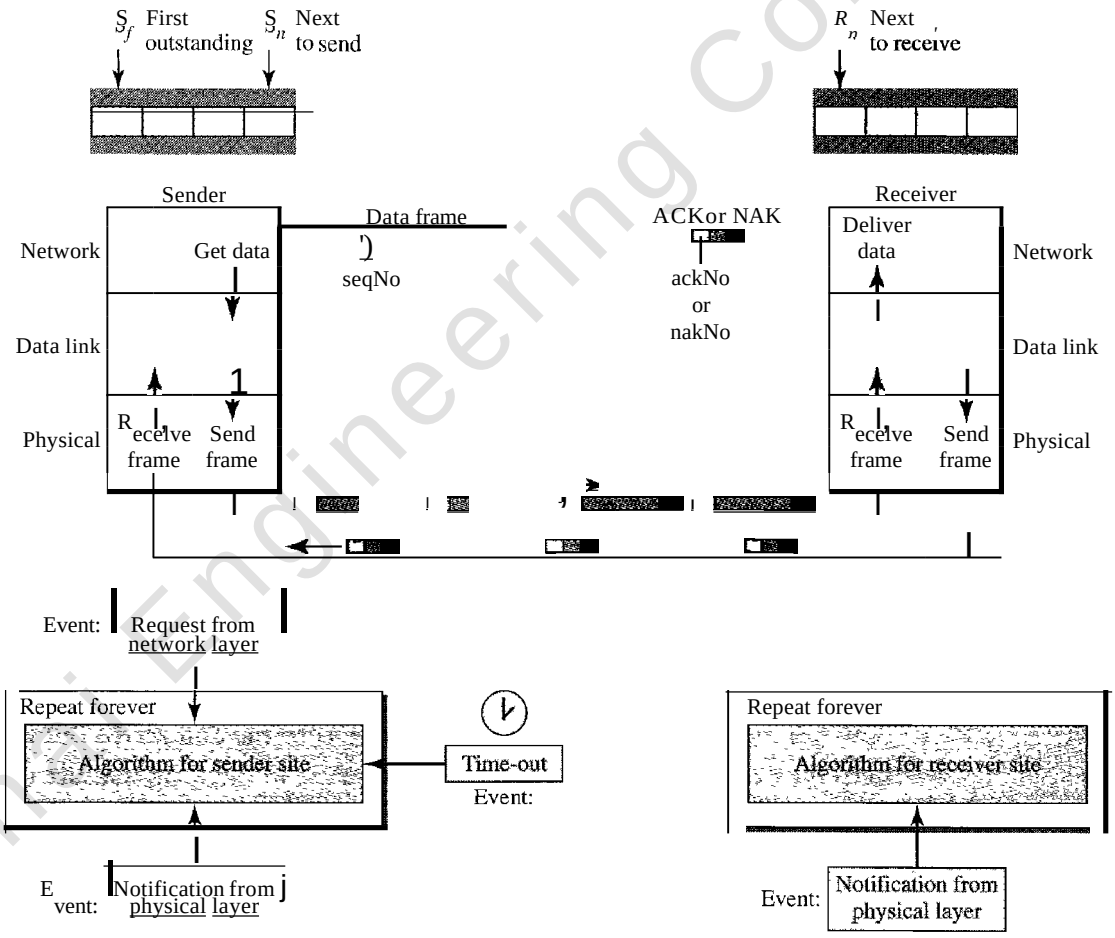


protocol. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

Design

The design in this case is to some extent similar to the one we described for the 00-Back-N, but more complicated, as shown in Figure 11.20.

Figure 11.20 *Design of Selective Repeat ARQ*

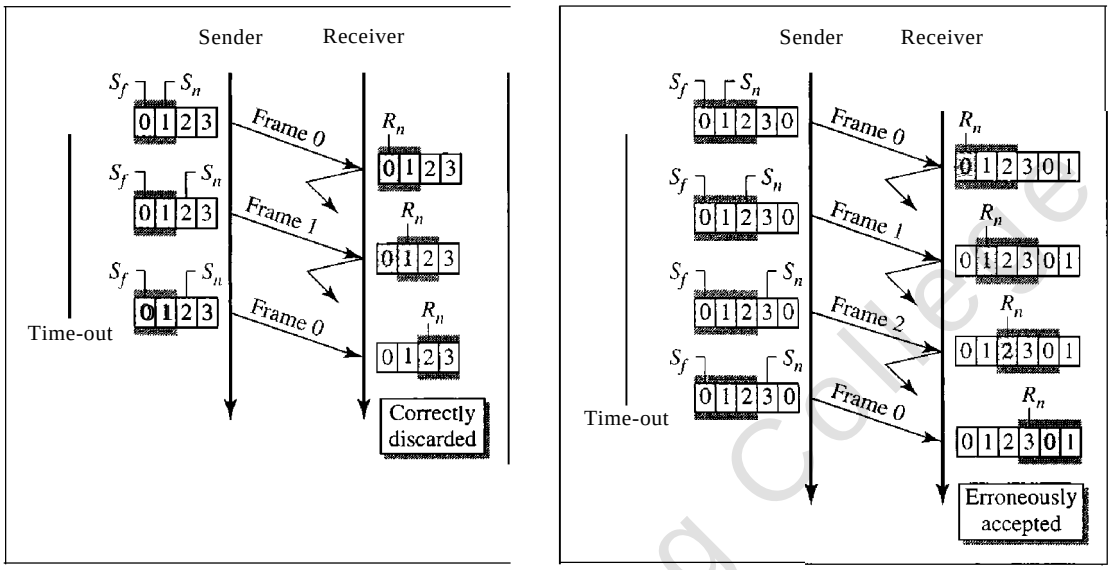


Window Sizes

We can now show why the size of the sender and receiver windows must be at most one-half of 2^m. For an example, we choose $m = 2$, which means the size of the window is $2^m/2$, or 2. Figure 11.21 compares a window size of 2 with a window size of 3.

If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent. However, the window of the receiver is now expecting

Figure 11.21 *Selective Repeat ARQ, window size*



a. Window size $= 2^{m} - 1$ b. Window size $> 2^{m} - 1$

frame 2, not frame 0, so this duplicate frame is correctly discarded. When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0. However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error.

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^{m}.

Algorithms

Algorithm 11.9 shows the procedure for the sender.

Algorithm 11.9 *Sender-site Selective Repeat algorithm*

```
1     = 2^m - 1 i
2     = Oi
3     = Oi
4
5   hile (true)                          //Repeat forever
6  {
7    WaitForEvent()i
8    if(Event(RequestToSend)}             //There is a packet to sen
9    {
```

Algorithm 11.9 *Sender-site Selective Repeat algorithm (continued)*

```
10      if(Sn-S;E >= Sw)                    I/If window is full
11            Sleep{);
12      GetData{) ;
13      MakeFrame(Sn) ;
14      StoreFrame{Sn);
15      SendFrame(Sn) ;
16      Sn = Sn + 1;
17      StartTimer{Sn);
18    }
19
20    if(Event{ArrivalNotification»       IIACK arrives
21    {
22      Receive{frame);                   I/Receive ACK or NAK
23      if{corrupted{frame»
24            Sleep();
25      if (FrameType == NAK)
26        if (nakNo between Sf and So)
27        {
28          resend{nakNo);
29          StartTimer{nakNo);
30        }
31      if (FrameType == ACK)
32        if (ackNo between Sf and So)
33        {
34          while{sf < ackNo)
35          {
36            Purge(sf);
37            stopTimer(Sf) ;
38            Sf = Sf + 1;
39          }
40        }
41    }
42
43    if(Event{TimeOut{t»)              liThe timer expires
44    {
45      StartTimer{t);
46      SendFrame{t);
47    }
48  }
```

Analysis The handling of the request event is similar to that of the previous protocol except that one timer is started for each frame sent. The arrival event is more complicated here. An ACK or a NAK frame may arrive. If a valid NAK frame arrives, we just resend the corresponding frame. If a valid ACK arrives, we use a loop to purge the buffers, stop the corresponding timer. and move the left wall of the window. The time-out event is simpler here; only the frame which times out is resent.

Algorithm 11.10 shows the procedure for the receiver.

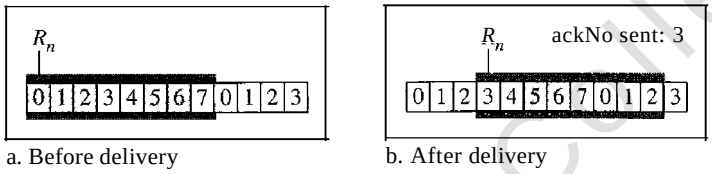
Algorithm **11.10** *Receiver-site Selective Repeat algorithm*

```
1  Rn = 0;
2  NakSent = false;
3  AckNeeded = false;
4  Repeat(for all slots)
5      Marked(slot) = false;
6
7  !while (true)                                  IIRepeat forever
8  {
9    WaitForEvent()i
10
11   if{Event{ArrivalNotification»                jData frame arrives
12   {
13       Receive(Frame);
14       if(corrupted(Frame»&& (NOT NakSent)
15       {
16        SendNAK(Rn);
17        NakSent = true;
18        Sleep{);
19       }
20       if(seqNo <> Rn)&& (NOT NakSent)
21       {
22        SendNAK(Rn);
23        NakSent = true;
24        if {(seqNo in window)&&(IMarked(seqNo»
25        {
26         StoreFrame{seqNo)
27         Marked(seqNo)= true;
28         whi1e(Marked(Rn)
29         {
30          DeliverData(Rn);
31          Purge(Rn);
32          Rn = Rn + 1;
33          AckNeeded = true;
34         }
35          if(AckNeeded);
36          {
37          SendAck(Rn);
38          AckNeeded = false;
39          NakSent = false;
40          }
41        }
42       }
43   }
44 }
```

Analysis Here we need more initialization. In order not to overwhelm the other side with NAKs, we use a variable called NakSent. To know when we need to send an ACK, we use a variable called AckNeeded. Both of these are initialized to false. We also use a set of variables to

mark the slots in the receive window once the corresponding frame has arrived and is stored. If we receive a corrupted frame and a NAK has not yet been sent, we send a NAK to tell the other site that we have not received the frame we expected. If the frame is not corrupted and the sequence number is in the window, we store the frame and mark the slot. If contiguous frames, starting from R_n have been marked, we deliver their data to the network layer and slide the window. Figure 11.22 shows this situation.

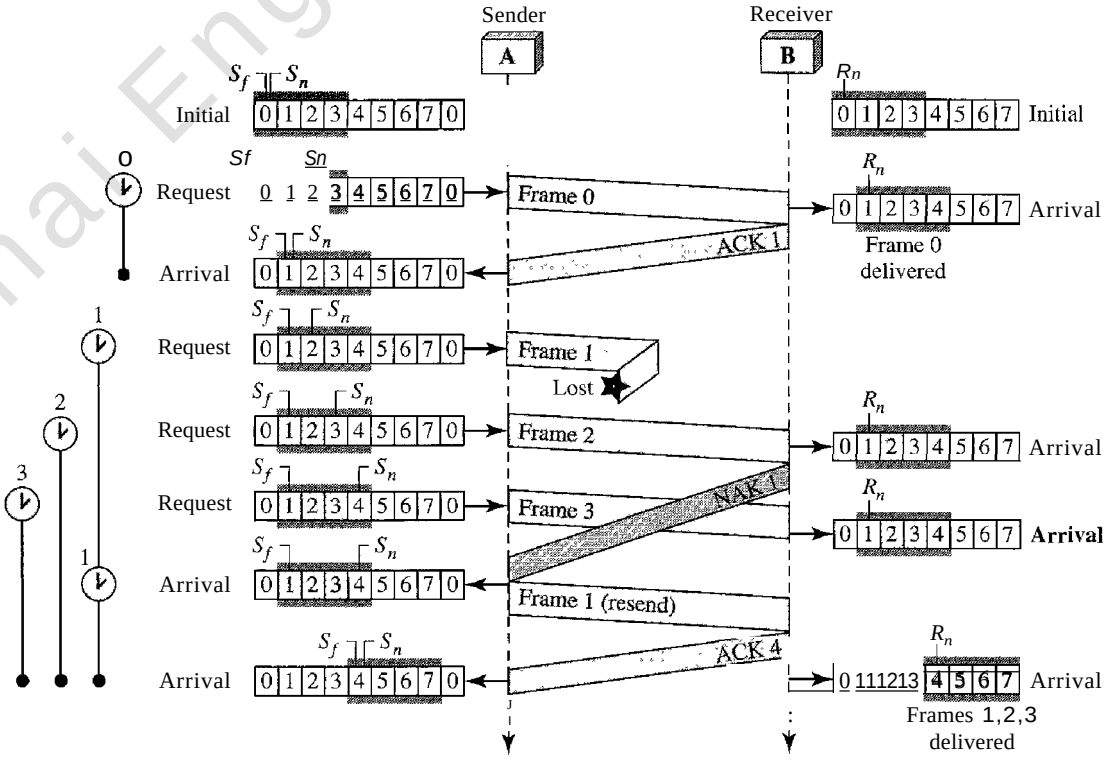
Figure 11.22 *Delivery of data in Selective Repeat ARQ*



Example 11.8

This example is similar to Example 11.3 in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation.

Figure 11.23 *Flow diagram for Example 11.8*



One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame I starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked (colored slot), but it cannot be delivered because frame I is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames I, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. After the first alTival, there was only one frame and it started from the beginning of the window. After the last arrival, there are three frames and the first one starts from the beginning of the window.

Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAKI to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.

The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to *n* frames are delivered in one shot, only one ACK is sent for all of them.

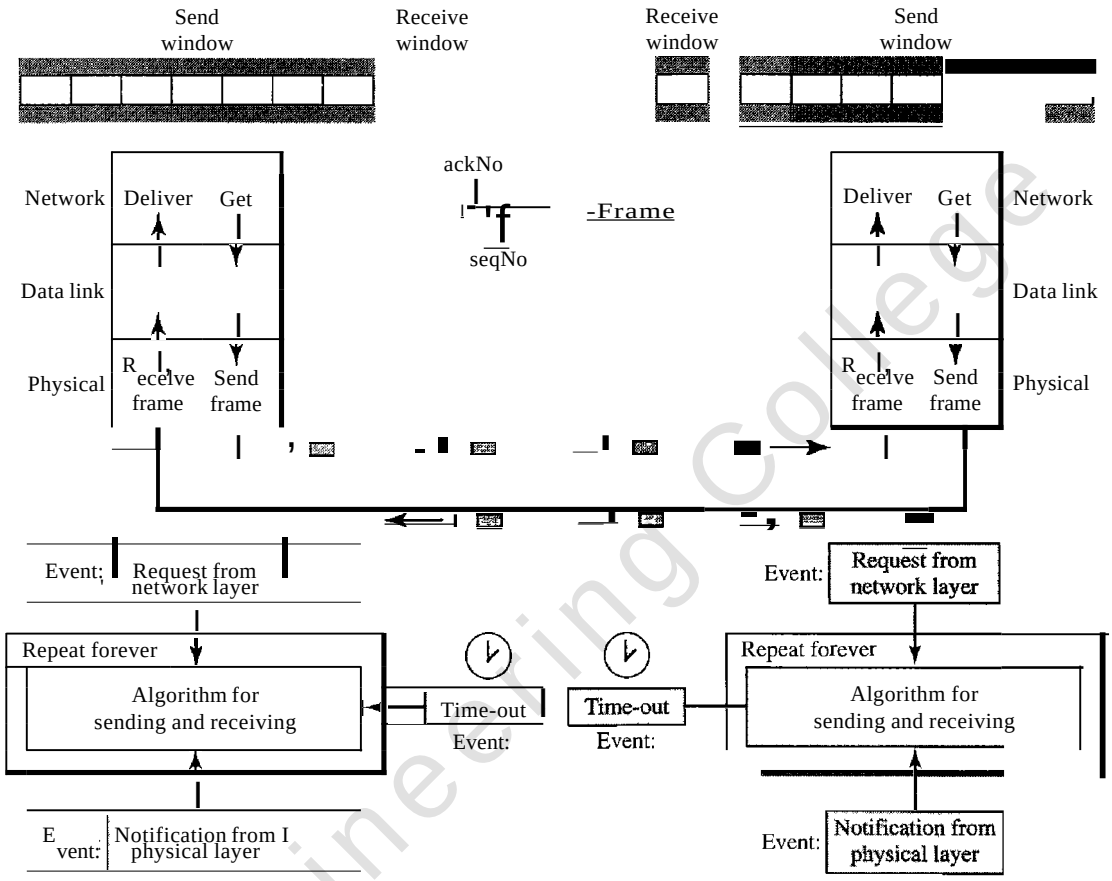
Piggybacking

The three protocols we discussed in this section are all unidirectional: data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction. In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

We show the design for a Go-Back-N ARQ using piggybacking in Figure 11.24. Note that each node now has two windows: one send window and one receive window. Both also need to use a timer. Both are involved in three types of events: request, arrival, and time-out. However, the arrival event here is complicated; when a frame arrives, the site needs to handle control information as well as the frame itself. Both of these concerns must be taken care of in one event, the arrival event. The request event uses only the send window at each site; the arrival event needs to use both windows.

An important point about piggybacking is that both sites must use the same algorithm. This algorithm is complicated because it needs to combine two arrival events into one. We leave this task as an exercise.

Figure 11.24 *Design ofpiggybacking in Go-Back-N ARQ*



11.6 HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms we discussed in this chapter.

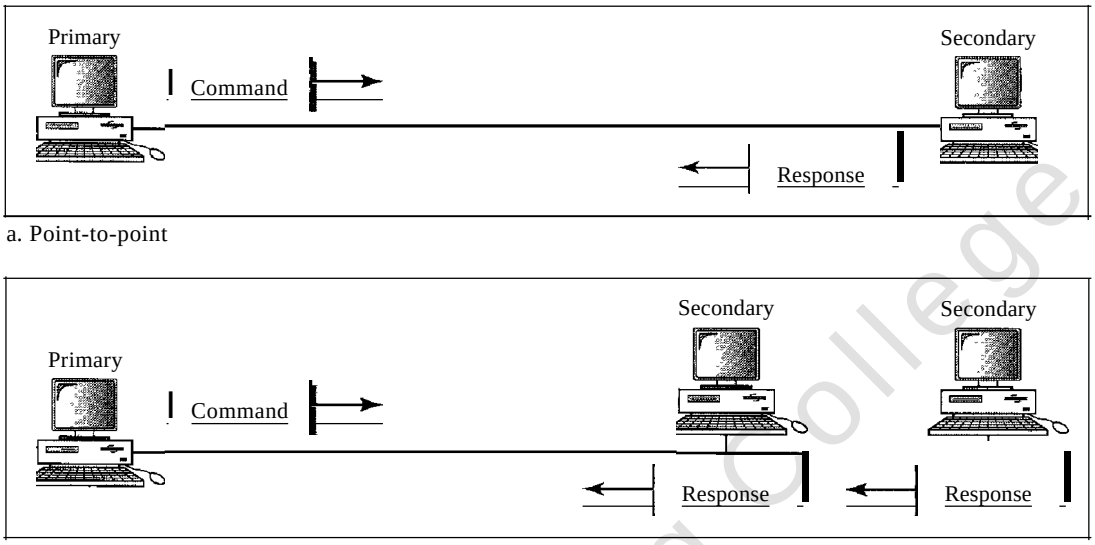
Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations: normal response mode (NRM) and asynchronous balanced mode (ABM).

Normal Response Mode

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands; a secondary station can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in Figure 11.25.

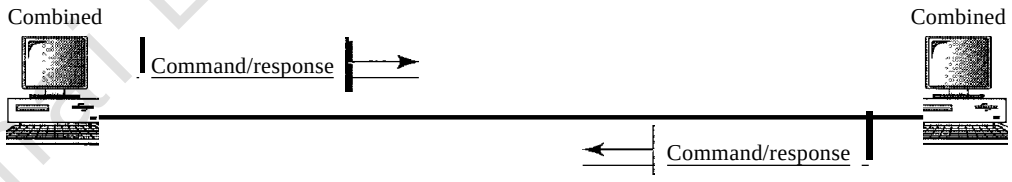
Figure 11.25 *Normal response mode*



Asynchronous Balanced Mode

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in Figure 11.26. This is the common mode today.

Figure 11.26 *Asynchronous balanced mode*



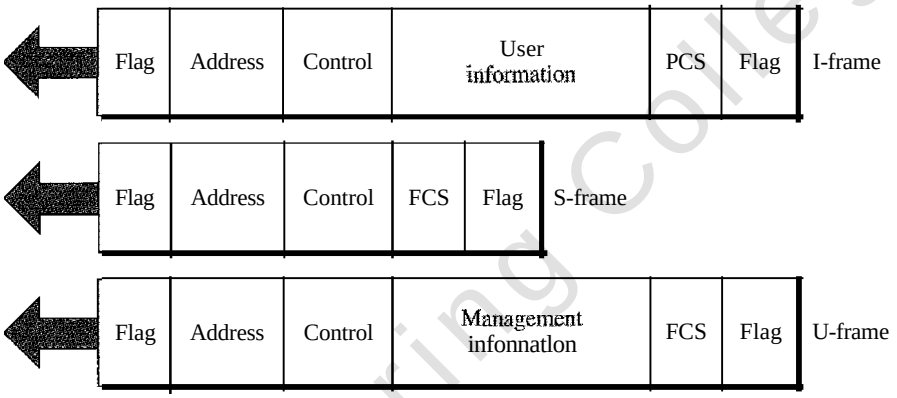
Frames

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: information frames (I-frames), supervisory frames (S-frames), and unnumbered frames (V-frames). Each type of frame serves as an envelope for the transmission of a different type of message. I-frames are used to transport user data and control information relating to user data (piggybacking). S-frames are used only to transport control information. V-frames are reserved for system management. Information carried by V-frames is intended for managing the link itself.

Frame Format

Each frame in HDLC may contain up to six fields, as shown in Figure 11.27: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

Figure 11.27 *HDLC frames*



Fields

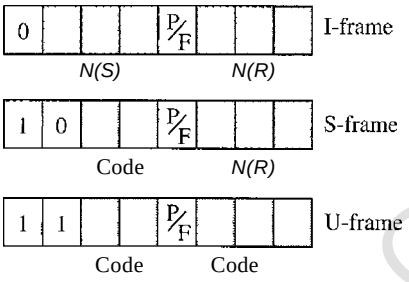
Let us now discuss the fields and their use in different frame types.

- Flag field. The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.
- Address field. The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary creates the frame, it contains *afrom* address. An address field can be 1 byte or several bytes long, depending on the needs of the network. One byte can identify up to 128 stations (1 bit is used for another purpose). Larger networks require multiple-byte address fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0; only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come.
- Control field. The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type. We discuss this field later and describe its format for each frame type.
- Information field. The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- FCS field. The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

Control Field

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in greater detail. The format is specific for the type of frame, as shown in Figure 11.28.

Figure 11.28 *Control field format for the different frame types*



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called *N(S),* define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7; but in the extension format, in which the control field is 2 bytes, this field is larger. The last 3 bits, called *N(R),* correspond to the acknowledgment number when piggybacking is used. The single bit between *N(S)* and *N(R)* is called the *PIF* bit. The *PIP* field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Fieldfor S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate (e.g., when the station either has no data of its own to send or needs to send a command or response other than an acknowledgment). S-frames do not have information fields. If the first 2 bits of the control field is 10, this means the frame is an S-frame. The last 3 bits, called *N(R),* corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame. The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

- Receive ready (RR). If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value *N(R)* field defines the acknowledgment number.

- Receive not ready (RNR). If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of *NCR)* is the acknowledgment number.
- Reject (REJ). If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in *Go-Back-N* ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of *NCR)* is the negative acknowledgment number.
- Selective reject (SREJ). If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat.* The value of *N(R)* is the negative acknowledgment number.

Control Fieldfor V-Frames

Unnumbered frames are used to exchange session management and control infonnation between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the infonnation carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the PtF bit and a 3-bit suffix after the PtF bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames. Some of the more common types are shown in Table 11.1.

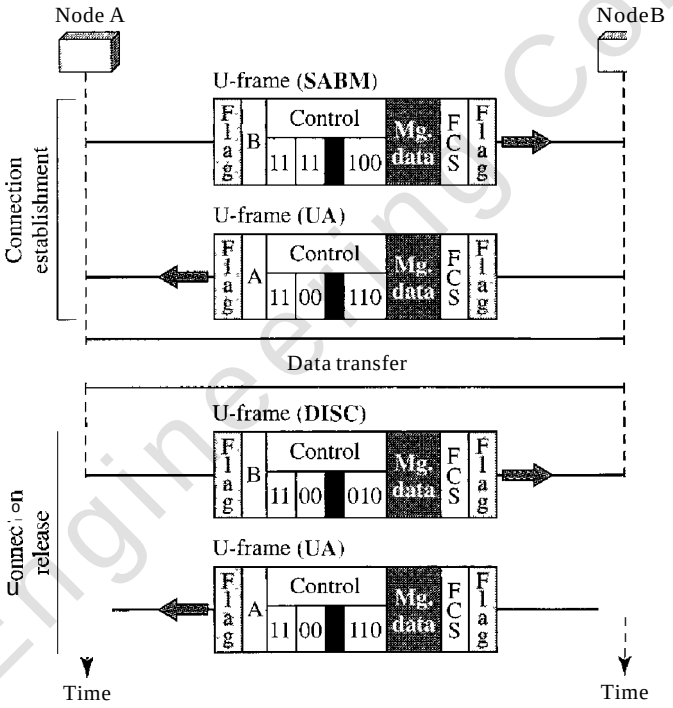
Table 11.1 *U-frame control command and response*

Code	*Command*	*Response*	*Meaning*
00 001	SNRM		Set normal response mode
11 011	SNRME		Set normal response mode, extended
11 100	SABM	DM	Set asynchronous balanced mode or disconnect mode
11110	SABME		Set asynchronous balanced mode, extended
00 000	UI	UI	Unnumbered information
00 110		UA	Unnumbered acknowledgment
00 010	DISC	RD	Disconnect or request disconnect
10 000	SIM	RIM	Set initialization mode or request information mode
00 100	UP		Unnumbered poll
11 001	RSET		Reset
11 101	XID	XID	Exchange ID
10 001	FRMR	FRMR	Frame reject

Example 11.9: *Connection/Disconnection*

Figure 11.29 shows how V-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (VA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a VA (unnumbered acknowledgment).

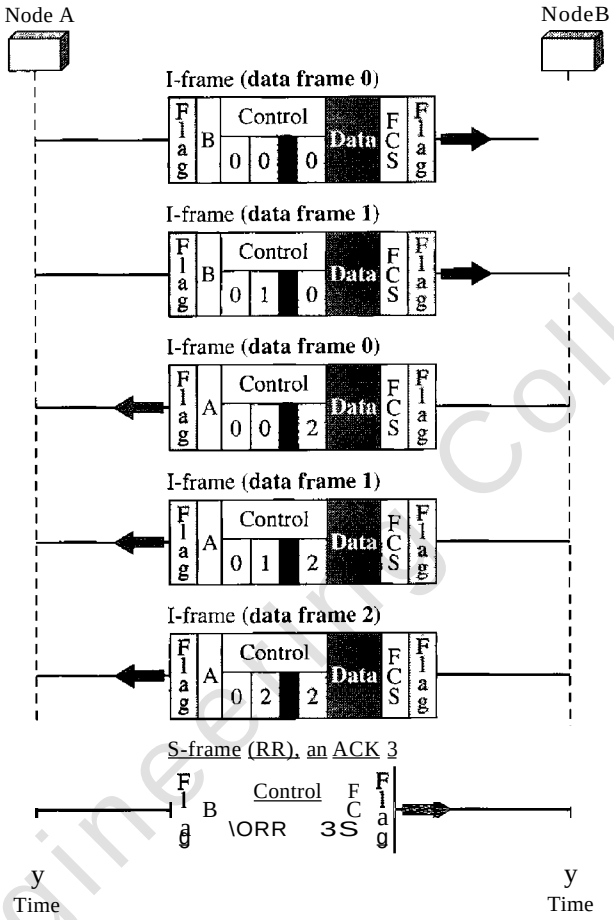
Figure **11.29** *Example of connection and disconnection*



Example 11.10: Piggybacking without Error

Figure 11.30 shows an exchange using piggybacking. Node A begins the exchange of information with an I-frame numbered 0 followed by another I-frame numbered 1. Node B piggybacks its acknowledgment of both frames onto an I-frame of its own. Node B's first I-frame is also numbered 0 *[N(S)* field] and contains a 2 in its *N(R)* field, acknowledging the receipt of A's frames 1 and 0 and indicating that it expects frame 2 to arrive next. Node B transmits its second and third I-frames (numbered 1 and 2) before accepting further frames from node A. Its *N(R)* information, therefore, has not changed: B frames 1 and 2 indicate that node B is still expecting A's frame 2 to arrive next. Node A has sent all its data. Therefore, it cannot piggyback an acknowledgment onto an I-frame and sends an S-frame instead. The RR code indicates that A is still ready to receive. The number 3 in the *N(R)* field tells B that frames 0, 1, and 2 have all been accepted and that A is now expecting frame number 3.

Figure 11.30 *Example ofpiggybacking without error*



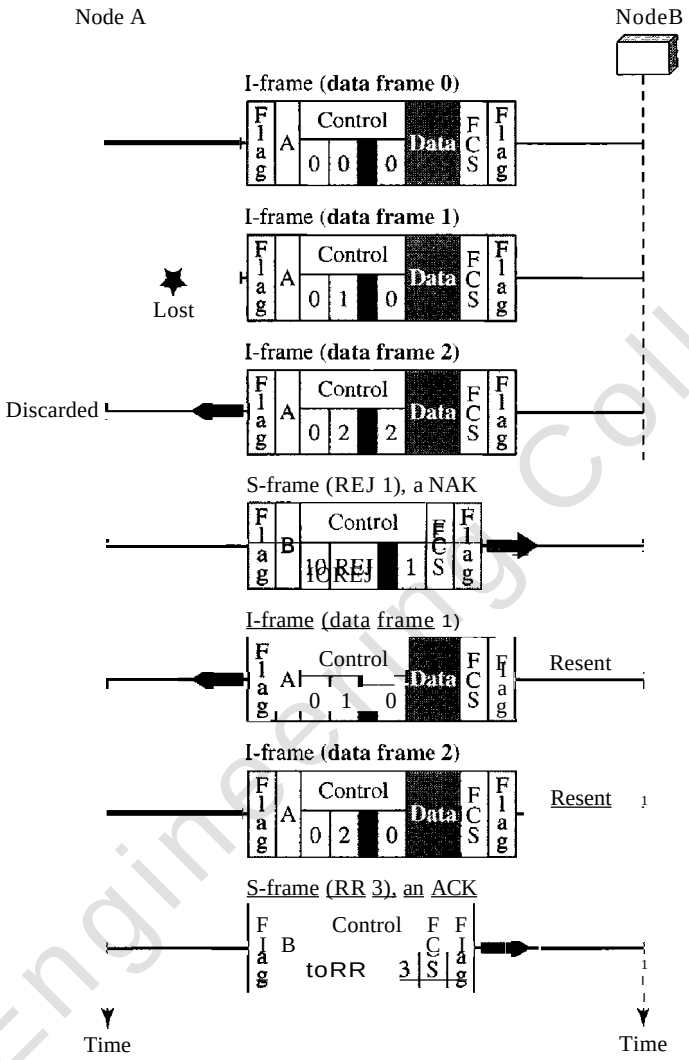
Example 11.11: *Piggybacking with Error*

Figure 11.31 shows an exchange in which a frame is lost. Node B sends three data frames (0, 1, and 2), but frame 1 is lost. When node A receives frame 2, it discards it and sends a *REI* frame for frame 1. Note that the protocol being used is *Go-Back-N* with the special use of an REI frame as a NAK frame. The NAK frame does two things here: It confirms the receipt of frame and declares that frame 1 and any following frames must be resent. Node B, after receiving the REI frame, resends frames 1 and 2. Node A acknowledges the receipt by sending an RR frame (ACK) with acknowledgment number 3.

11.7 POINT-TO-POINT PROTOCOL

Although HDLC is a general protocol that can be used for both point-to-point and multi-point configurations, one of the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP). Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and

Figure 11.31 *Example of piggybacking with error*



manage the transfer of data, there is a need for a point-to-point protocol at the data link layer. PPP is by far the most common.

PPP provides several services:

1. PPP defines the format of the frame to be exchanged between devices.
2. PPP defines how two devices can negotiate the establishment of the link and the exchange of data.
3. PPP defines how network layer data are encapsulated in the data link frame.
4. PPP defines how two devices can authenticate each other.
5. PPP provides multiple network layer services supporting a variety of network layer protocols.
6. PPP provides connections over multiple links.
7. PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

On the other hand, to keep PPP simple, several services are missing:

1. PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.
2. PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order.
3. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

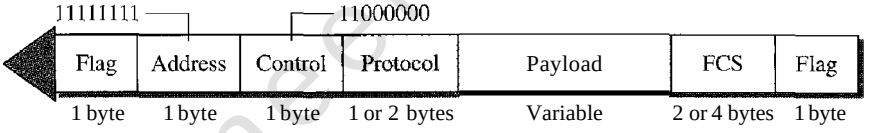
Framing

PPP is a byte-oriented protocol. Framing is done according to the discussion of byte-oriented protocols at the beginning of this chapter.

Frame Format

Figure 11.32 shows the format of a PPP frame. The description of each field follows:

Figure 11.32 *PPP frame format*



- Flag. A PPP frame starts and ends with a I-byte flag with the bit pattern 01111110. Although this pattern is the same as that used in HDLC, there is a big difference. PPP is a byte-oriented protocol; HDLC is a bit-oriented protocol. The flag is treated as a byte, as we will explain later.
- Address. The address field in this protocol is a constant value and set to 11111111 (broadcast address). During negotiation (discussed later), the two parties may agree to omit this byte.
- Control. This field is set to the constant value 11000000 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection. This means that this field is not needed at all, and again, the two parties can agree, during negotiation, to omit this byte.
- Protocol. The protocol field defines what is being carried in the data field: either user data or other information. We discuss this field in detail shortly. This field is by default 2 bytes long, but the two parties can agree to use only I byte.
- Payload field. This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte-stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.
- FCS. The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRe.

Byte Stuffing

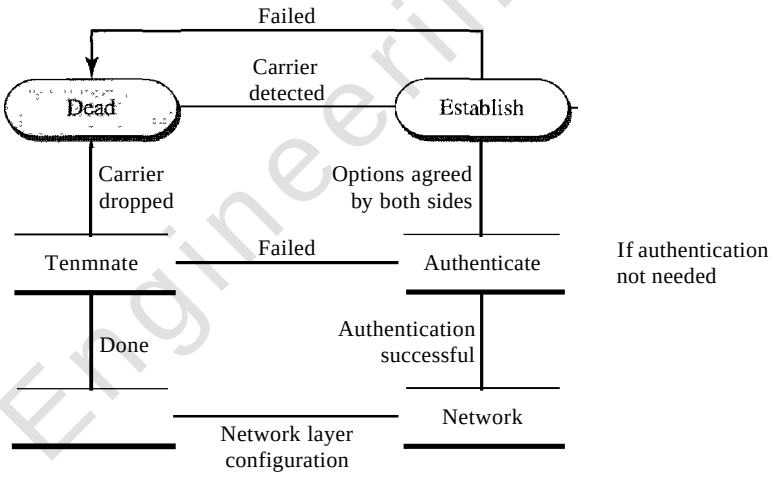
The similarity between PPP and HDLe ends at the frame format. PPP, as we discussed before, is a byte-oriented protocol totally different from HDLC. As a byte-oriented protocol, the flag in PPP is a byte and needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that every time the flaglike pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag.

PPP is a byte-oriented protocol using byte stuffing with the escape byte 01111101.

Transition Phases

A PPP connection goes through phases which can be shown in a transition phase diagram (see Figure 11.33).

Figure 11.33 *Transition phases*



- Dead. In the dead phase the link is not being used. There is no active carrier (at the physical layer) and the line is quiet.
- Establish. When one of the nodes starts the communication, the connection goes into this phase. In this phase, options are negotiated between the two parties. If the negotiation is successful, the system goes to the authentication phase (if authentication is required) or directly to the networking phase. The link control protocol packets, discussed shortly, are used for this purpose. Several packets may be exchanged here.
- Authenticate. The authentication phase is optional; the two nodes may decide, during the establishment phase, not to skip this phase. However, if they decide to proceed with authentication, they send several authentication packets, discussed later. If the result is successful, the connection goes to the networking phase; otherwise, it goes to the termination phase.
- Network. In the network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at

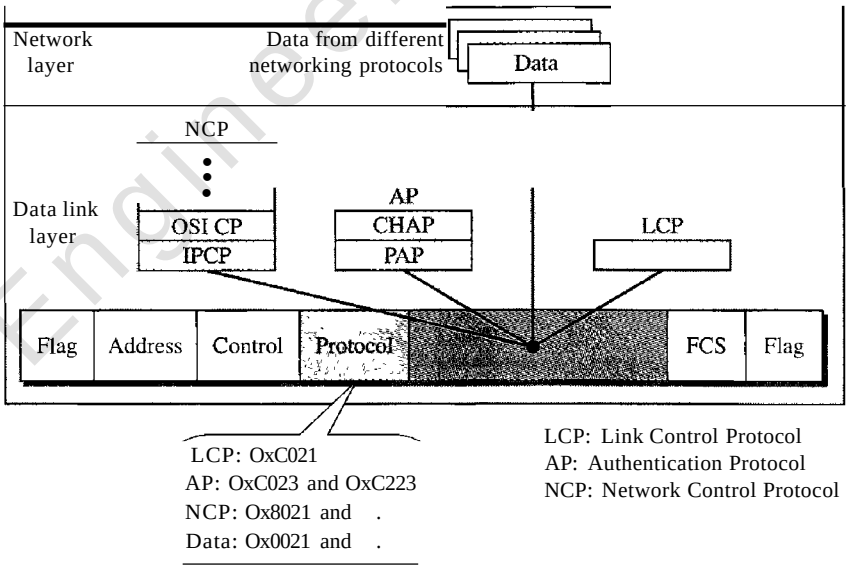
the network layer can be exchanged. The reason is that PPP supports multiple protocols at the network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive the data.

- **Open.** In the open phase, data transfer takes place. When a connection reaches this phase, the exchange of data packets can be started. The connection remains in this phase until one of the endpoints wants to terminate the connection.
- **Terminate.** In the termination phase the connection is terminated. Several packets are exchanged between the two ends for house cleaning and closing the link.

Multiplexing

Although PPP is a data link layer protocol, PPP uses another set of other protocols to establish the link, authenticate the parties involved, and carry the network layer data. Three sets of protocols are defined to make PPP powetful: the Link Control Protocol (LCP), two Authentication Protocols (APs), and several Network Control Protocols (NCPs). At any moment, a PPP packet can carry data from one of these protocols in its data field, as shown in Figure 11.34. Note that there is one LCP, two APs, and several NCPs. Data may also come from several different network layers.

Figure **11.34** *Multiplexing in PPP*



Link Control Protocol

The **Link Control Protocol** (LCP) is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established. See Figure 11.35.

All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal.

The code field defines the type of LCP packet. There are 11 types of packets as shown in Table 11.2.

Figure 11.35 *LCP packet encapsulated in aframe*

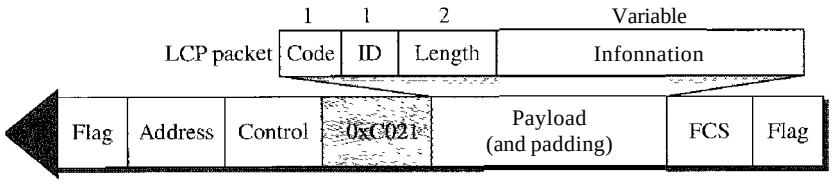


Table 11.2 *LCP packets*

Code	*Packet Type*	*Description*
OxOl	Coofigure-request	Contains the list of proposed options and their values
Ox02	Configure-ack	Accepts all options proposed
Ox03	Configure-nak	Announces that some options are oot acceptable
Ox04	Configure-reject	Announces that some options are not recognized
Ox05	Terminate-request	Request to shut down the line
Ox06	Terminate-ack	Accept the shutdown request
Ox07	Code-reject	Announces an unknown code
Ox08	Protocol-reject	Announces an unknown protocol
Ox09	Echo-request	A type of hello message to check if the other end is alive
OxOA	Echo-reply	The response to the echo-request message
OxOB	Discard-request	A request to discard the packet

There are three categories of packets. The first category, comprising the first four packet types, is used for link configuration during the establish phase. The second category, comprising packet types 5 and 6, is used for link tennination during the termination phase. The last five packets are used for link monitoring and debugging.

The ID field holds a value that matches a request with a reply. One endpoint inserts a value in this field, which will be copied into the reply packet. The length field defines the length of the entire LCP packet. The information field contains information, such as options, needed for some LCP packets.

There are many options that can be negotiated between the two endpoints. Options are inserted in the information field of the configuration packets. In this case, the information field is divided into three fields: option type, option length, and option data. We list some of the most common options in Table 11.3.

Table 11.3 *Common options*

Option	*Default*
Maximum receive unit (payload field size)	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

Authentication Protocols

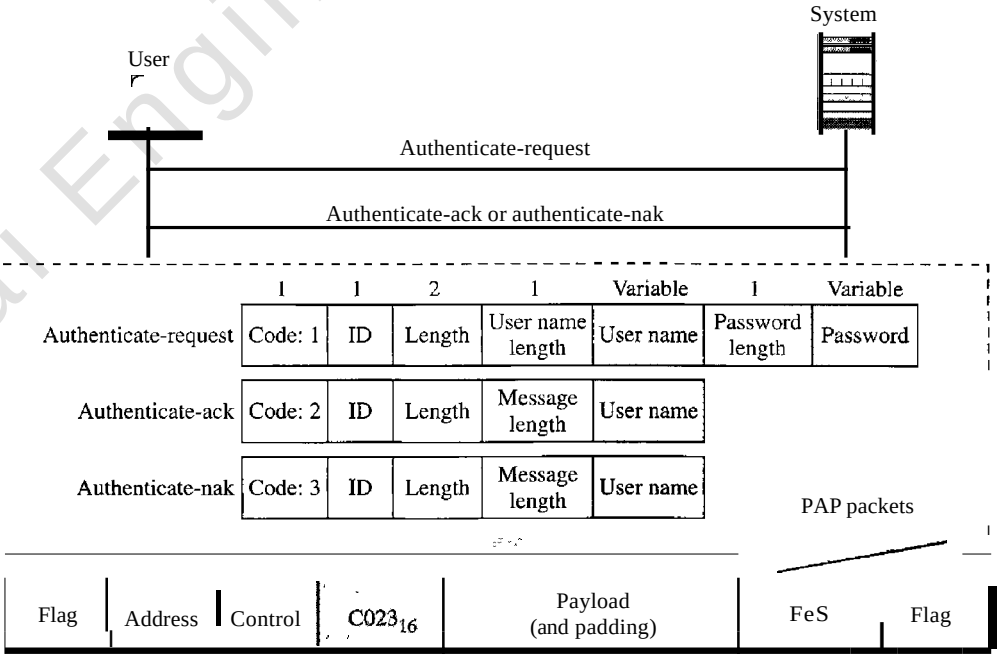
Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. **Authentication** means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

1. The user who wants to access a system sends an authentication identification (usually the user name) and a password.
2. The system checks the validity of the identification and password and either accepts or denies connection.

Figure 11.36 shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is OxC023. The three PAP packets are authenticate-request, authenticate-ack, and authenticate-nak. The first packet is used by the user to send the user name and password. The second is used by the system to allow access. The third is used by the system to deny access.

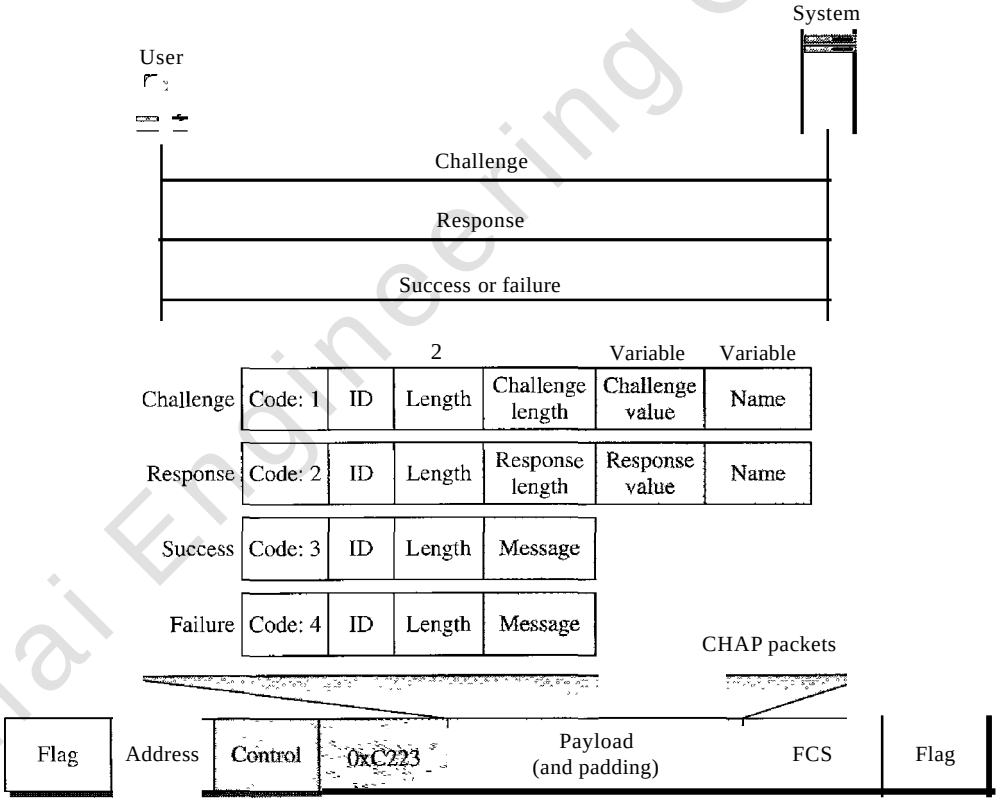
Figure 11.36 *PAP packets encapsulated in a PPP frame*



CHAP The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way hand-shaking authentication protocol that provides greater security than PAP. **In** this method, the password is kept secret; it is never sent online.

1. The system sends the user a challenge packet containing a challenge value, usually a few bytes.
2. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.
3. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret. Figure 11.37 shows the packets and how they are used.

Figure 11.37 *CHAP packets encapsulated in a PPP frame*



CHAP packets are encapsulated in the PPP frame with the protocol value C223 in hexadecimal. There are four CHAP packets: challenge, response, success, and failure. The first packet is used by the system to send the challenge value. The second is used by the user to return the result of the calculation. The third is used by the system to allow access to the system. The fourth is used by the system to deny access to the system.

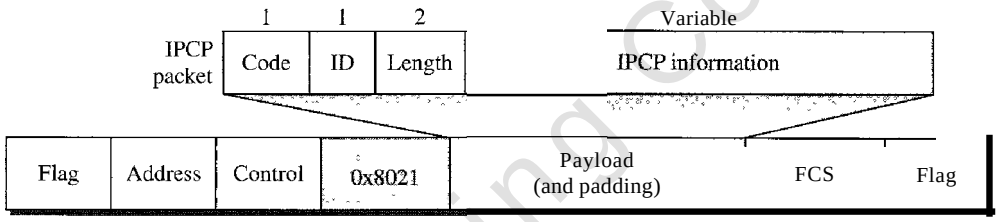
Network Control Protocols

PPP is a multiple-network layer protocol. It can carry a network layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on.

To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets. Xerox CP does the same for the Xerox protocol data packets, and so on. Note that none of the NCP packets carry network layer data; they just configure the link at the network layer for the incoming data.

IPCP One NCP protocol is the Internet Protocol Control Protocol (IPCP). This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 11.38. Note that the value of the protocol field in hexadecimal is 8021.

Figure 11.38 *fPCP packet encapsulated in PPP frame*



IPCP defines seven packets, distinguished by their code values, as shown in Table 11.4.

Table 11.4 *Code value for IPCP packets*

Code	*IPCP Packet*
OxO!	Configure-request
Ox02	Configure-ack
Ox03	Configure-nak
Ox04	Configure-reject
Ox05	Terminate-request
Ox06	Terminate-ack
Ox07	Code-reject

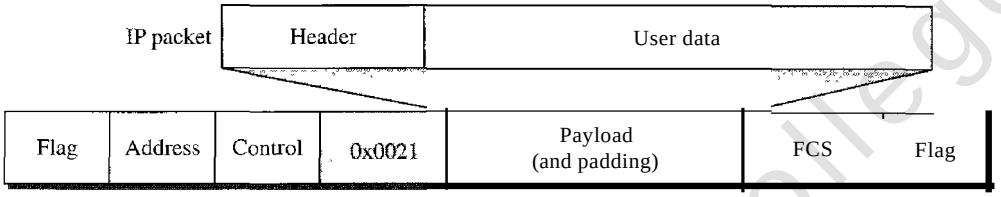
Other Protocols There are other NCP protocols for other network layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on. The value of the code and the format of the packets for these other protocols are the same as shown in Table 11.4.

Data/rom the Network Layer

After the network layer configuration is completed by one of the NCP protocols, the users can exchange data packets from the network layer. Here again, there are different

protocol fields for different network layers. For example, if PPP is carrying data from the IP network layer, the field value is 0021 (note that the three rightmost digits are the same as for IPCP). If PPP is carrying data from the OSI network layer, the value of the protocol field is 0023, and so on. Figure 11.39 shows the frame for IP.

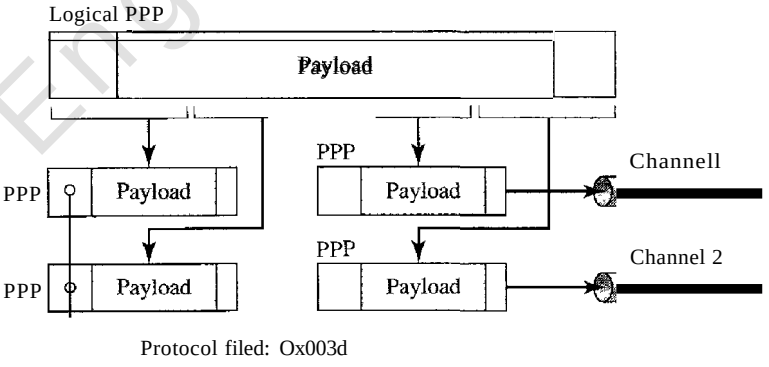
Figure 11.39 *IP datagram encapsulated in a PPP frame*



Multilink PPP

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 11.40. To show that the actual PPP frame is carrying a fragment of a

Figure **11.40** *Multilink PPP*

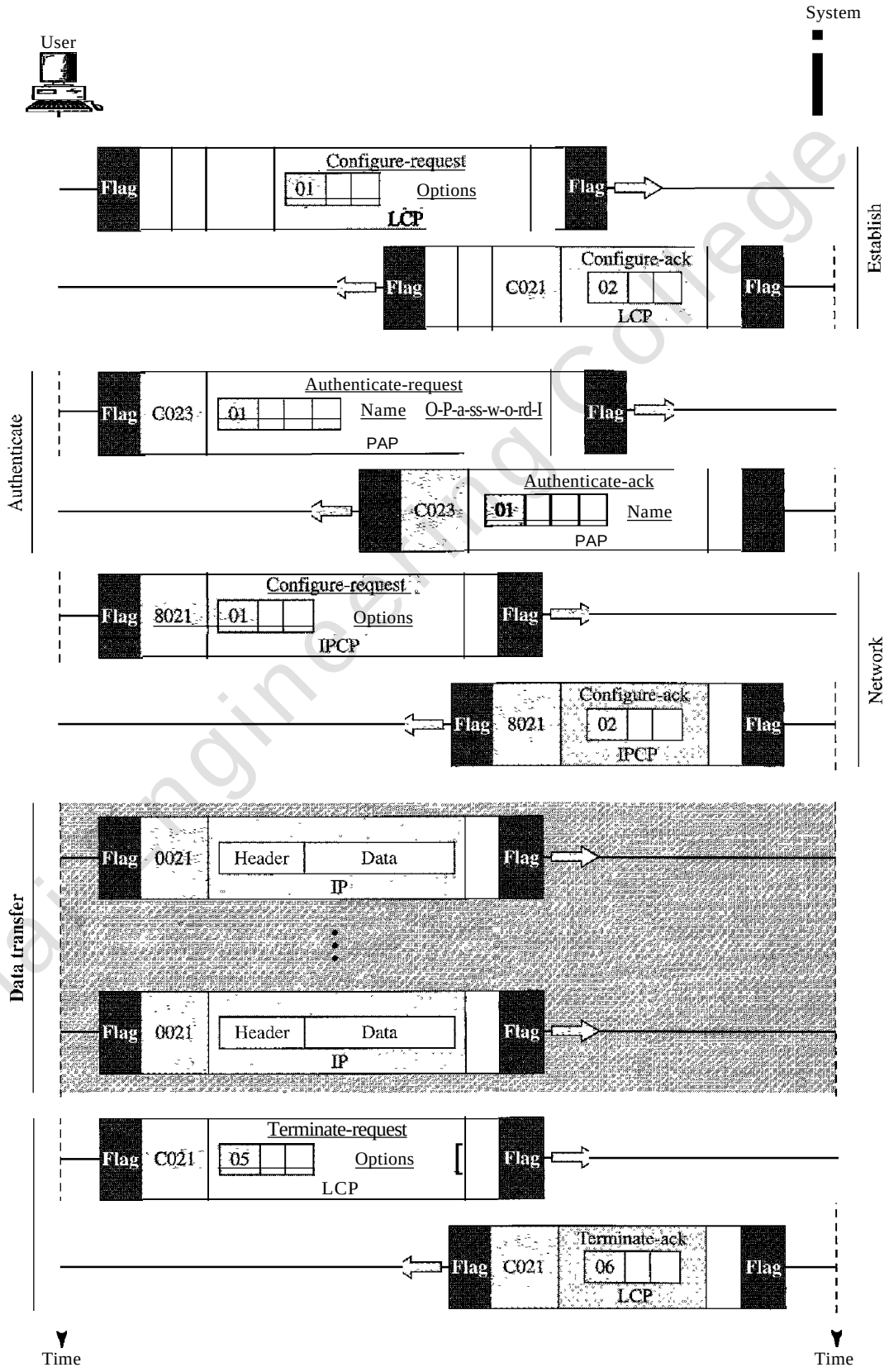


logical PPP frame, the protocol field is set to Ox003d. This new development adds complexity. For example, a sequence number needs to be added to the actual PPP frame to show a fragment's position in the logical frame.

Example 11.12

Let us go through the phases followed by a network layer packet as it is transmitted through a PPP connection. Figure 11.41 shows the steps. For simplicity, we assume unidirectional movement of data from the user site to the system site (such as sending an e-mail through an ISP).

Figure 11.41 *An example*



The first two frames show link establishment. We have chosen two options (not shown in the figure): using PAP for authentication and suppressing the address control fields. Frames 3 and 4 are for authentication. Frames 5 and 6 establish the network layer connection using IPCP.

The next several frames show that some IP packets are encapsulated in the PPP frame. The system (receiver) may have been running several network layer protocols, but it knows that the incoming data must be delivered to the IP protocol because the NCP protocol used before the data transfer was IPCP.

After data transfer, the user then terminates the data link connection, which is acknowledged by the system. Of course the user or the system could have chosen to terminate the network layer IPCP and keep the data link layer running if it wanted to run another NCP protocol.

The example is trivial, but it points out the similarities of the packets in LCP, AP, and NCP. It also shows the protocol field values and code numbers for particular protocols.

11.8 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

A discussion of data link control can be found in [GW04], Chapter 3 of [Tan03], Chapter 7 of [Sta04], Chapter 12 of [Kes97], and Chapter 2 of [PD03]. More advanced materials can be found in [KMK04].

11.9 KEY TERMS

acknowledgment (ACK)
asynchronous balanced mode (ABM)
automatic repeat request (ARQ)
bandwidth-delay product
bit-oriented protocol
bit stuffing
byte stuffing
Challenge Handshake Authentication Protocol (CHAP)
character-oriented protocol
data link control
error control
escape character (ESC)
event
fixed-size framing
flag
flow control
framing
Go-Back-N ARQ Protocol
High-level Data Link Control (HDLC)
information frame (I-frame)
Internet Protocol Control Protocol (IPCP)
Link Control Protocol (LCP)
negative acknowledgment (NAK)
noiseless channel
noisy channel
normal response mode (NRM)
Password Authentication Protocol (PAP)
piggybacking
pipelining
Point-to-Point Protocol (PPP)
primary station

receive sliding window
secondary station
Selective Repeat ARQ Protocol
send sliding window
sequence number
Simplest Protocol
sliding window
Stop-and-Wait ARQ Protocol
Stop-and-Wait Protocol
supervisory frame (S-frame)
transition phase
unnumbered frame (D-frame)
variable-size framing

11.10 SUMMARY

- Data link control deals with the design and procedures for communication between two adjacent nodes: node-to-node communication.
- Framing in the data link layer separates a message from one source to a destination, or from other messages going from other sources to other destinations,
- Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of frames; in variable-size framing, we need a delimiter (flag) to define the boundary of two frames.
- Variable-size framing uses two categories of protocols: byte-oriented (or character-oriented) and bit-oriented. In a byte-oriented protocol, the data section of a frame is a sequence of bytes; in a bit-oriented protocol, the data section of a frame is a sequence of bits.
- In byte-oriented (or character-oriented) protocols, we use byte stuffing; a special byte added to the data section of the frame when there is a character with the same pattern as the flag.
- In bit-oriented protocols, we use bit stuffing; an extra 0 is added to the data section of the frame when there is a sequence of bits with the same pattern as the flag.
- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment. Error control refers to methods of error detection and correction.
- For the noiseless channel, we discussed two protocols: the Simplest Protocol and the Stop-and-Wait Protocol. The first protocol has neither flow nor error control; the second has no error control. In the Simplest Protocol, the sender sends its frames one after another with no regards to the receiver. In the Stop-and-Wait Protocol, the sender sends one frame, stops until it receives confirmation from the receiver, and then sends the next frame.
- For the noisy channel, we discussed three protocols: Stop-and-Wait ARQ, 00-*Back-N,* and Selective Repeat ARQ. The Stop-and-Wait ARQ Protocol, adds a simple error control mechanism to the Stop-and-Wait Protocol. In the *Oo-Back-N* ARQ Protocol, we can send several frames before receiving acknowledgments, improving the efficiency of transmission. In the Selective Repeat ARQ protocol we avoid unnecessary transmission by sending only frames that are corrupted.
- Both *Oo-Back-N* and Selective-Repeat Protocols use a sliding window. In 00-*Back-N* ARQ, if m is the number of bits for the sequence number, then the size of

the send window must be less than 2^m; the size of the receiver window is always 1. In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

- A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about frames from B; when a frame is carrying data from B to A, it can also carry control information about frames from A.
- High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. However, the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP), which is a byte-oriented protocol.

11.11 PRACTICE SET

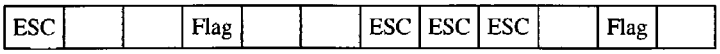
Review Questions

1. Briefly describe the services provided by the data link layer.
2. Define framing and the reason for its need.
3. Compare and contrast byte-oriented and bit-oriented protocols. Which category has been popular in the past (explain the reason)? Which category is popular now (explain the reason)?
4. Compare and contrast byte-stuffing and bit-stuffing. Which technique is used in byte-oriented protocols? Which technique is used in bit-oriented protocols?
5. Compare and contrast flow control and error control.
6. What are the two protocols we discussed for noiseless channels in this chapter?
7. What are the three protocols we discussed for noisy channels in this chapter?
8. Explain the reason for moving from the Stop-and-Wait ARQ Protocol to the 00-Back-N ARQ Protocol.
9. Compare and contrast the Go-Back-N ARQ Protocol with Selective-RepeatARQ.
10. Compare and contrast HDLC with PPP. Which one is byte-oriented; which one is bit-oriented?
11. Define piggybacking and its usefulness.
12. Which of the protocols described in this chapter utilize pipelining?

Exercises

13. Byte-stuff the data in Figure 11.42.

Figure 11.42 *Exercise 13*



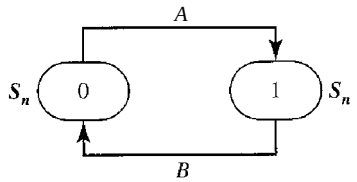
14. Bit-stuff the data in Figure 11.43.

Figure 11.43 *Exercise 14*

10001111111001111101000111111111111000011111

15. Design two simple algorithms for byte-stuffing. The first adds bytes at the sender; the second removes bytes at the receiver.
16. Design two simple algorithms for bit-stuffing. The first adds bits at the sender; the second removes bits at the receiver.
17. A sender sends a series of packets to the same destination using 5-bit sequence numbers. If the sequence number starts with 0, what is the sequence number after sending 100 packets?
18. Using 5-bit sequence numbers, what is the maximum size of the send and receive windows for each of the following protocols?
    a. Stop-and-Wait ARQ
    b. Go-Back-N ARQ
    c. Selective-Repeat ARQ
19. Design a bidirectional algorithm for the Simplest Protocol using piggybacking. Note that the both parties need to use the same algorithm.
20. Design a bidirectional algOIithm for the Stop-and-Wait Protocol using piggybacking. Note that both parties need to use the same algorithm.
21. Design a bidirectional algorithm for the Stop-and-Wait ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
22. Design a bidirectional algorithm for the Go-Back-N ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
23. Design a bidirectional algorithm for the Selective-Repeat ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
24. Figure 11.44 shows a state diagram to simulate the behavior of Stop-and-Wait ARQ at the sender site.

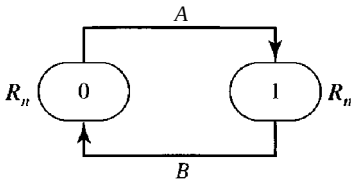
Figure 11.44 *Exercise 24*



The states have a value of *Sn* (0 or 1). The arrows shows the transitions. Explain the events that cause the two transitions labeled A and B.

25. Figure 11.45 shows a state diagram to simulate the behavior of Stop-and-Wait ARQ at the receiver site.

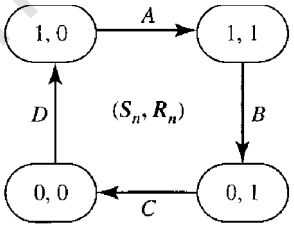
Figure 11.45 *Exercise 25*



The states have a value of R_n (0 or 1). The arrows shows the transitions. Explain the events that cause the two transitions labeled A and B.

26. In Stop-and-Wait ARQ, we can combine the state diagrams of the sender and receiver in Exercises 24 and 25. One state defines the combined values of R_n and S_n. This means that we can have four states, each defined by *(x, y),* where *x* defines the value of *Sn* and *y* defines the value of R_n. In other words, we can have the four states shown in Figure 11.46. Explain the events that cause the four transitions labeled A, B, C, and D.

Figure 11.46 *Exercise 26*



27. The timer of a system using the Stop-and-Wait ARQ Protocol has a time-out of 6 ms. Draw the flow diagram similar to Figure 11.11 for four frames if the round trip delay is 4 ms. Assume no data frame or control frame is lost or damaged.
28. Repeat Exercise 27 if the time-out is 4 ms and the round trip delay is 6.
29. Repeat Exercise 27 if the first frame (frame 0) is lost.
30. A system uses the Stop-and-Wait ARQ Protocol. If each packet carries 1000 bits of data, how long does it take to send 1 million bits of data if the distance between the sender and receiver is 5000 KIn and the propagation speed is 2×10^8 m? Ignore transmission, waiting, and processing delays. We assume no data or control frame is lost or damaged.
31. Repeat Exercise 30 using the *Go-back-N* ARQ Protocol with a window size of 7. Ignore the overhead due to the header and trailer.
32. Repeat Exercise 30 using the Selective-Repeat ARQ Protocol with a window size of 4. Ignore the overhead due to the header and the trailer.

CHAPTER 12

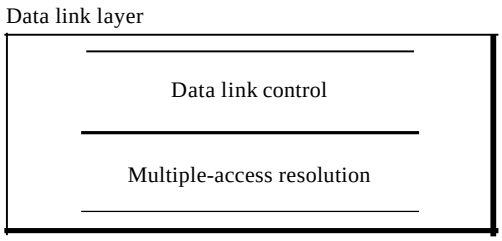
Multiple Access

In Chapter 11 we discussed data link control, a mechanism which provides a link with reliable communication. In the protocols we described, we assumed that there is an available dedicated link (or channel) between the sender and the receiver. This assumption mayor may not be true. If, indeed, we have a dedicated link, as when we connect to the Internet using PPP as the data link control protocol, then the assumption is true and we do not need anything else.

On the other hand, if we use our cellular phone to connect to another cellular phone, the channel (the band allocated to the vendor company) is not dedicated. A person a few feet away from us may be using the same channel to talk to her friend.

We can consider the data link layer as two sublayers. The upper sublayer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media. If the channel is dedicated, we do not need the lower sublayer. Figure 12.1 shows these two sublayers in the data link layer.

Figure 12.1 *Data link layer divided into two functionality-oriented sublayers*



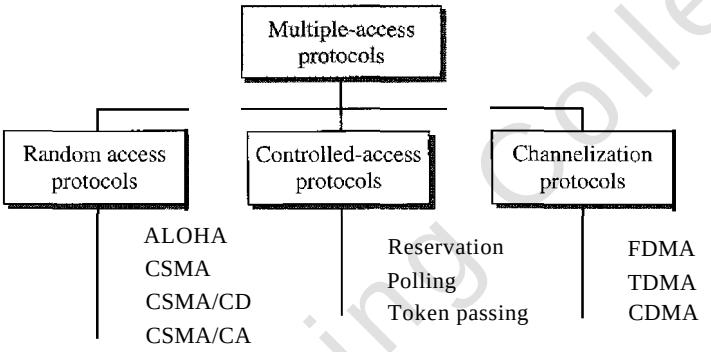
We will see in Chapter 13 that the IEEE has actually made this division for LANs. The upper sublayer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sublayer that is mostly responsible for multiple-access resolution is called the media access control (MAC) layer.

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking

in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, do not monopolize the discussion, and so on.

The situation is similar for multipoint networks. Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups. Protocols belonging to each group are shown in Figure 12.2.

Figure 12.2 *Taxonomy of multiple-access protocols discussed in this chapter*



12.1 RANDOM ACCESS

In random access or contention methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including the testing of the state of the medium.

Two features give this method its name. First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access.* Second, no rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict-collision-and the frames will be either destroyed or modified. To avoid access conflict or to resolve it when it happens, each station follows a procedure that answers the following questions:

- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

The random access methods we study in this chapter have evolved from a very interesting protocol known as ALOHA, which used a very simple procedure called multiple access (MA). The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called carrier sense multiple access. This method later evolved into two parallel methods: carrier sense multiple access with collision detection (CSMAlCD) and carrier sense multiple access with collision avoidance *(CSMA/CA)*. *CSMA/CD* tells the station what to do when a collision is detected. *CSMA/CA* tries to avoid the collision.

ALOHA

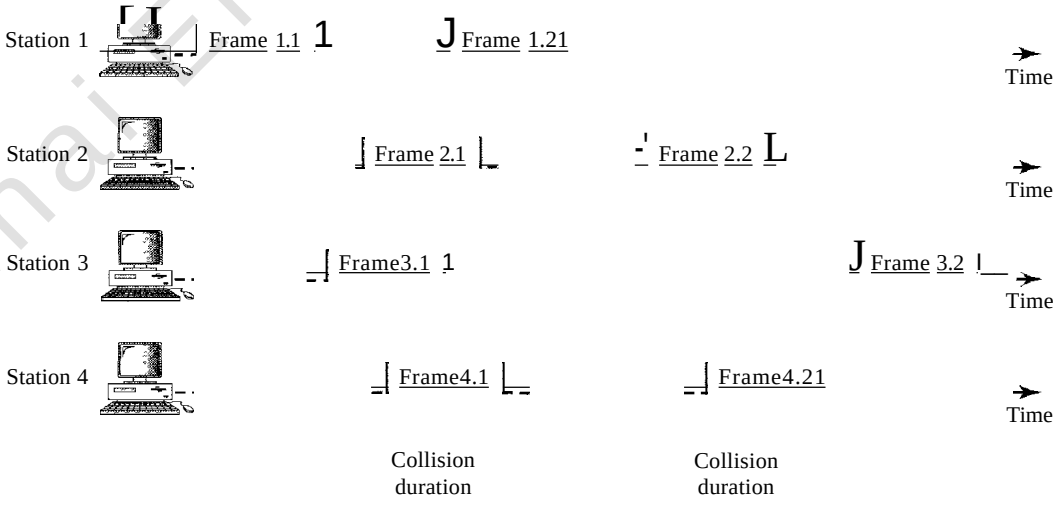
ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

Pure ALOHA

The original ALOHA protocol is called pure ALOHA. This is a simple, but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send. However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure 12.3 shows an example of frame collisions in pure ALOHA.

Figure 12.3 *Frames in a pure ALOHA network*



There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.3 shows that only

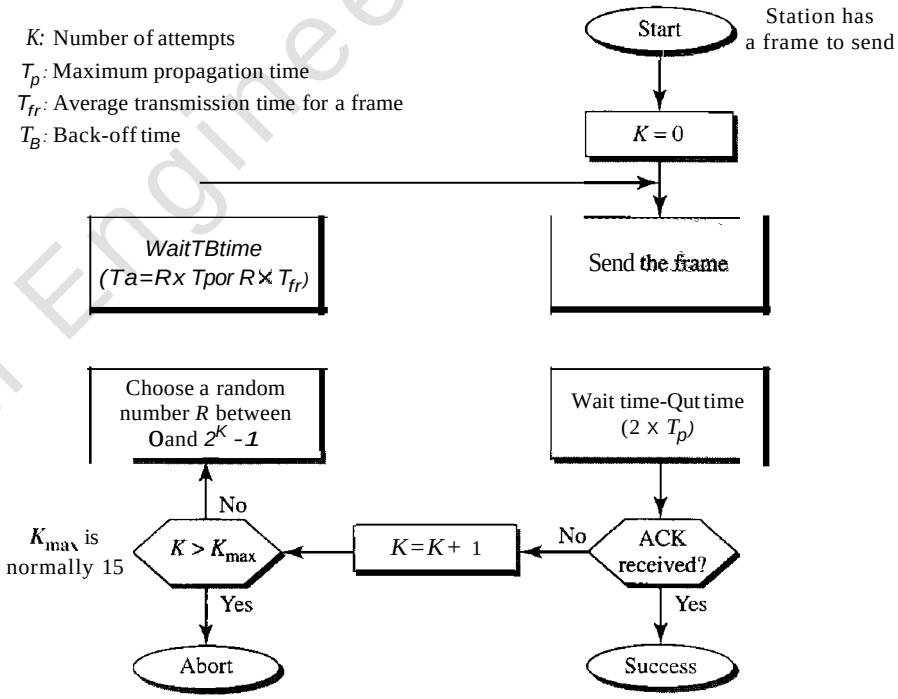
two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}, a station must give up and try later. Figure 12.4 shows the procedure for pure ALOHA based on the above strategy.

Figure **12.4** *Procedure for pure ALOHA protocol*



The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations (2 x T_p)' The back-off time T_B is a random value that normally depends on K (the number of attempted unsuccessful transmissions). The formula for T_B depends on the implementation. One common formula is the **binary exponential back-off.** In this

method, for each retransmission, a multiplier in the range 0 to $2^K - 1$ is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find T_B. Note that in this procedure, the range of the random numbers increases after each collision. The value of K_{max} is usually chosen as 15.

Example ***12.1***

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, we find $T_p = (600 \times 10^5) / (3 \times 10^8) = 2$ ms. Now we can find the value of T_B for different values of K.

a. For $K = 1$, the range is {0, 1}. The station needs to generate a random number with a value of 0 or 1. This means that T_B is either 0 ms (0 x 2) or 2 ms (1 x 2), based on the outcome of the random variable.

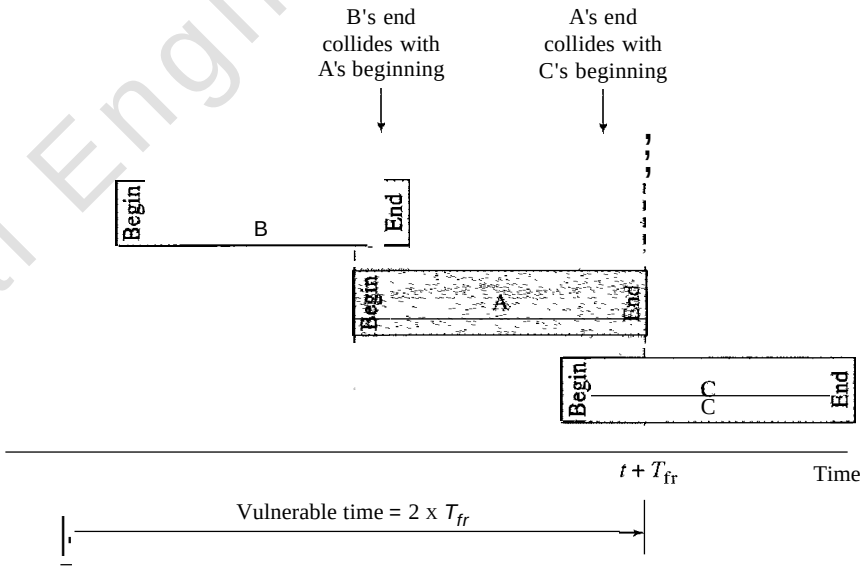
b. For $K = 2$, the range is {0, 1, 2, 3}. This means that T_B can be 0, 2, 4, or 6 ms, based on the outcome of the random variable.

c. For $K = 3$, the range is {0, 1, 2, 3, 4, 5, 6, 7}. This means that T_B can be 0, 2, 4, ... , 14 ms, based on the outcome of the random variable.

d. We need to mention that if $K > 10$, it is normally set to 10.

Vulnerable time Let us find the length of time, the **vulnerable time,** in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} s to send. Figure 12.5 shows the vulnerable time for station A.

Figure 12.5 *Vulnerable time for pure ALOHA protocol*



Station A sends a frame at time t. Now imagine station B has already sent a frame between $t - T_{fr}$ and t. This leads to a collision between the frames from station A and station B. The end of B's frame collides with the beginning of A's frame. On the other hand, suppose that station C sends a frame between t and $t + T_{fr}$. Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

Looking at Figure 12.5, we see that the vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

$$\text{Pure ALOHA vulnerable time} = 2 \times T_{fr}$$

Example 12.2

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

Solution

Average frame transmission time T_{fr} is 200 bits/200 kbps or 1 ms. The vulnerable time is 2×1 ms $= 2$ ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the one I-ms period that this station is sending.

Throughput Let us call G the average number of frames generated by the system during one frame transmission time. Then it can be proved that the average number of successful transmissions for pure ALOHA is $S = G \times e^{-2G}$. The maximum throughput S_{max} is 0.184, for $G = \frac{1}{2}$. In other words, if one-half a frame is generated during one frame transmission time (in other words, one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully. This is an expected result because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 0.184$ when $G = (1/2)$.

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

Solution

The frame transmission time is *2001200* kbps or 1 ms.

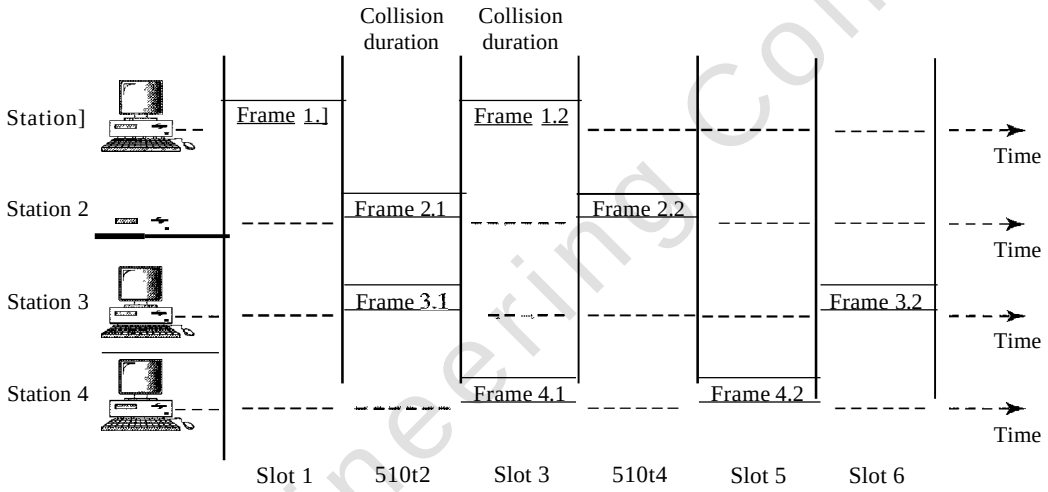
a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case $S = G \times e^{-2G}$ or $S = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.
b. If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is *(112).* In this case $S = G \times e^{-2G}$ or $S = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentagewise.
c. If the system creates 250 frames per second, this is (1/4) frame per millisecond. The load is (1/4). In this case $S = G \times e^{-2G}$ or $S = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Slotted ALOHA

Pure ALOHA has a vulnerable time of 2 x T_{fr}. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.

In slotted ALOHA we divide the time into slots of T_{fr} s and force the station to send only at the beginning of the time slot. Figure 12.6 shows an example of frame collisions in slotted ALOHA.

Figure 12.6 *Frames in a slotted ALOHA network*



Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to T_{fr} Figure 12.7 shows the situation.

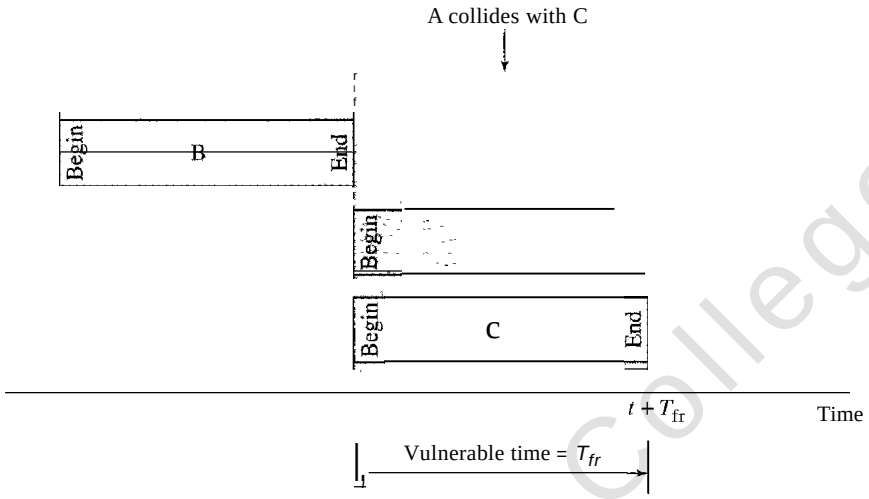
Figure 12.7 shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

$$\text{Slotted ALOHA vulnerable time} = T_{fr}$$

Throughput It can be proved that the average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$. The maximum throughput Smax is 0.368, when G = 1. In other words, if a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. This result can be expected because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when $G = 1$.

Figure 12.7 *Vulnerable time for slotted ALOHA protocol*



Example 12.4

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

Solution

This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is *200/200* kbps or 1 ms.

a. In this case G is 1. So S = G x e^{-G} or $S = 0.368$ (36.8 percent). This means that the throughput is 1000 x 0.0368 = 368 frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
b. Here G is $\frac{1}{2}$. In this case S = G x e^{-G} or S = 0.303 (30.3 percent). This means that the throughput is 500 x 0.0303 = 151. Only 151 frames out of 500 will probably survive.
c. Now Gis $\frac{1}{4}$. In this case S = G x e^{-G} or S = 0.195 (19.5 percent). This means that the throughput is 250 x 0.195 = 49. Only 49 frames out of 250 will probably survive.

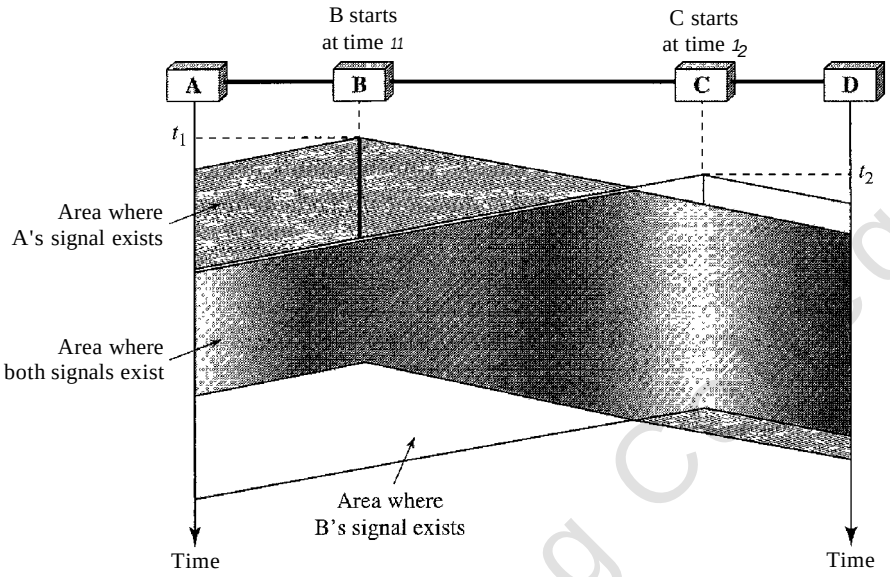
Carrier Sense Multiple Access (CSMA)

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk."

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 12.8, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station

Figure 12.8 *Space/time model of the collision in CSMA*



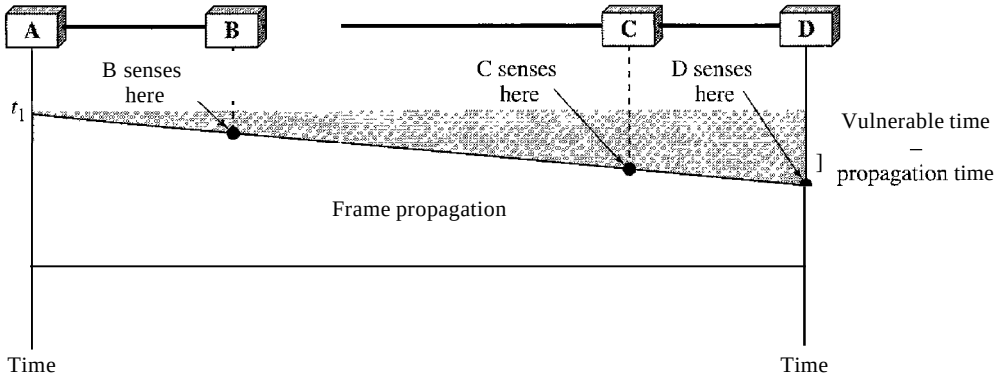
and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

At time t_1, station B senses the medium and finds it idle, so it sends a frame. At time t_2 ($t_2 > t_1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the propagation time T_p. This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 12.9 shows the worst

Figure 12.9 *Vulnerable time in CSMA*



case. The leftmost station A sends a frame at time t_1 which reaches the rightmost station D at time $t_1 + T_p$. The gray area shows the vulnerable area in time and space.

Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the I-persistent method, the nonpersistent method, and the p-persistent method. Figure 12.10 shows the behavior of three persistence methods when a station finds a channel busy.

Figure 12.10 *Behavior of three persistence methods*

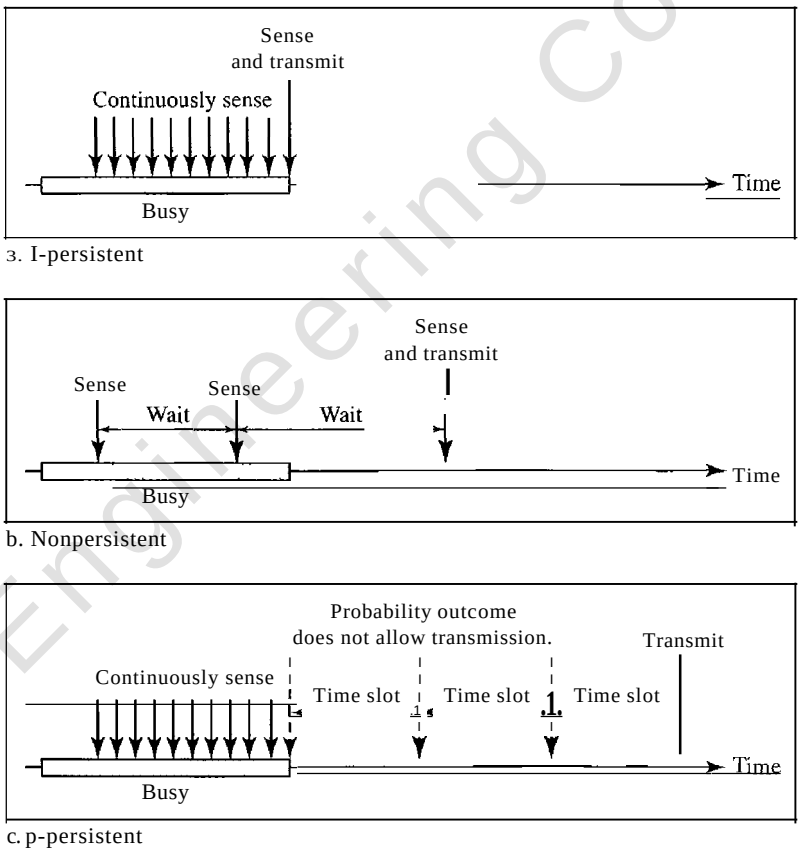
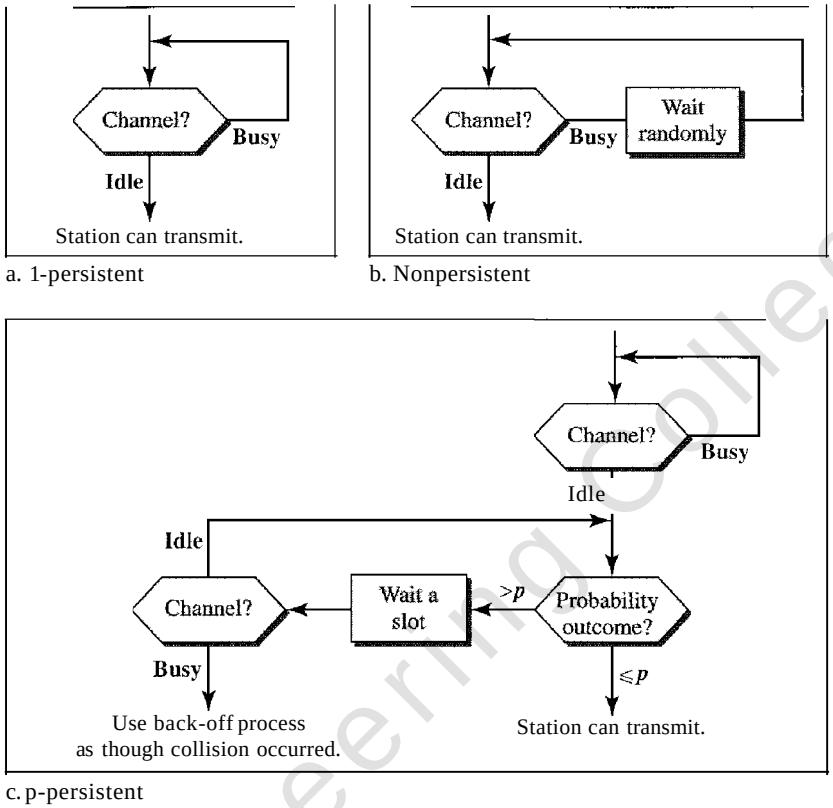


Figure 12.11 shows the flow diagrams for these methods.

I-Persistent The **I-persistent method** is simple and straightforward. **In** this method, after the station finds the line idle, it sends its frame immediately (with probability I). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately. We will see in Chapter 13 that Ethernet uses this method.

Nonpersistent **In** the **nonpersistent method,** a station that has a frame to send senses the line. If the line is idle, it sends immediately. **If** the line is not idle, it waits a

Figure 12.11 *Flow diagram for three persistence methods*



random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

p-Persistent **The p-persistent method** is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability p, the station sends its frame.
2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.
   a. If the line is idle, it goes to step 1.
   b. If the line is busy, it acts as though a collision has occurred and uses the back-off procedure.

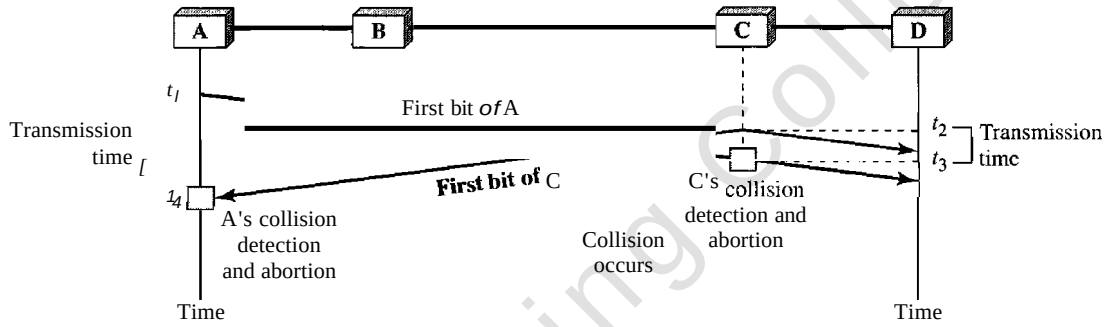
Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

The CSMA method does not specify the procedure following a collision. Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 12.12, stations A and C are involved in the collision.

Figure 12.12 *Collision of the first bit in CSMA/CD*



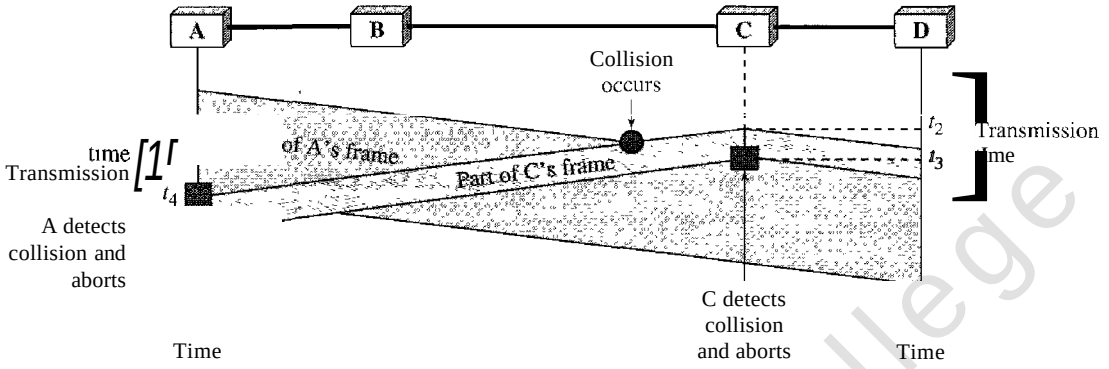
At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t_2. Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time t_4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration $t_4 - t_1$; C transmits for the duration $t_3 - t_2$. Later we show that, for the protocol to work, the length of any frame divided by the bit rate in this protocol must be more than either of these durations. At time t_4, the transmission of A's frame, though incomplete, is aborted; at time t_3, the transmission of B's frame, though incomplete, is aborted.

Now that we know the time durations for the two transmissions, we can show a more complete graph in Figure 12.13.

Minimum Frame Size

For *CSMA/CD* to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time T_{fr} must be at least two times the maximum propagation time T_p. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time T_p to reach the second, and the effect of the collision takes another time T_p to reach the first. So the requirement is that the first station must still be transmitting after $2T_p$.

Figure 12.13 *Collision and abortion in CSMA/CD*



Example 12.5

A network using *CSMA/CD* has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

Solution

The frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps x 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see in Chapter 13.

Procedure

Now let us look at the flow diagram for *CSMA/CD* in Figure 12.14. It is similar to the one for the ALOHA protocol, but there are differences.

The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes we discussed previously (nonpersistent, 1-persistent, or p-persistent). The corresponding box can be replaced by one of the persistence processes shown in Figure 12.11.

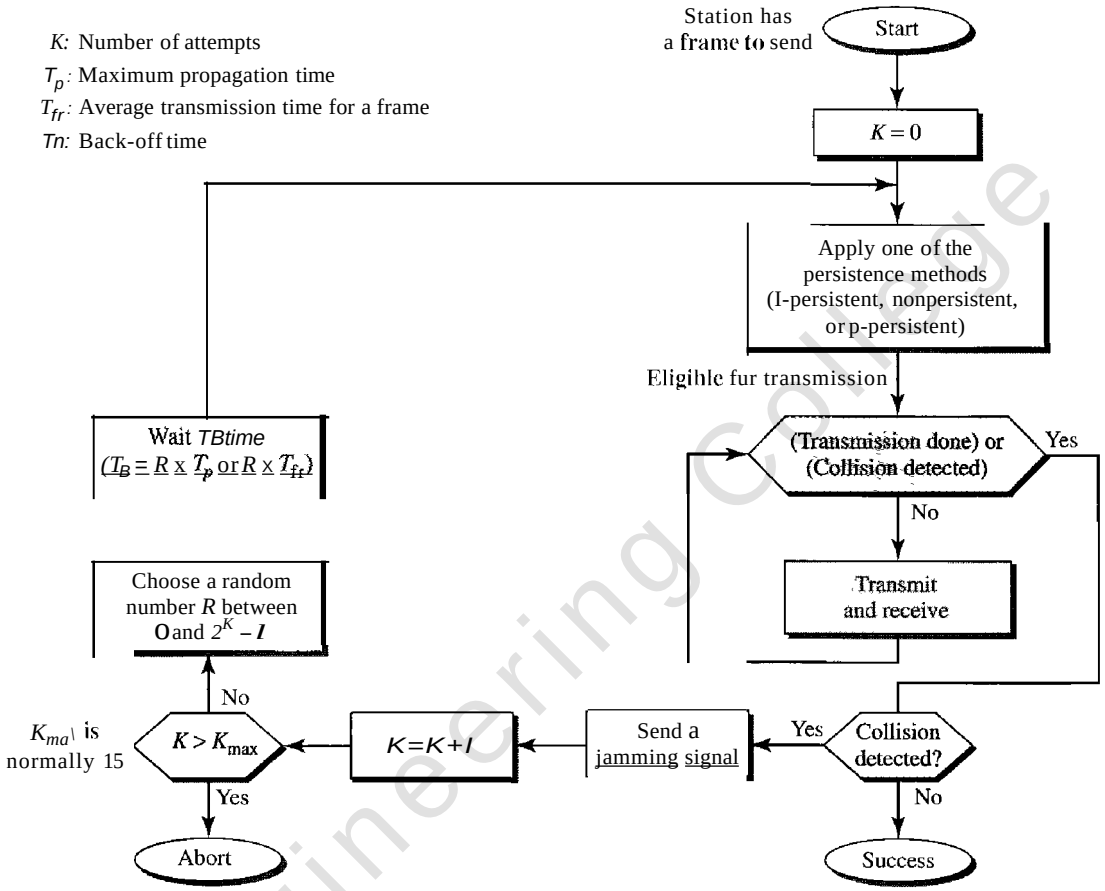
The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In *CSMA/CD*, transmission and collision detection is a continuous process. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports). We use a loop to show that transmission is a continuous process. We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.

The third difference is the sending of a short jamming signal that enforces the collision in case other stations have not yet sensed the collision.

Energy Level

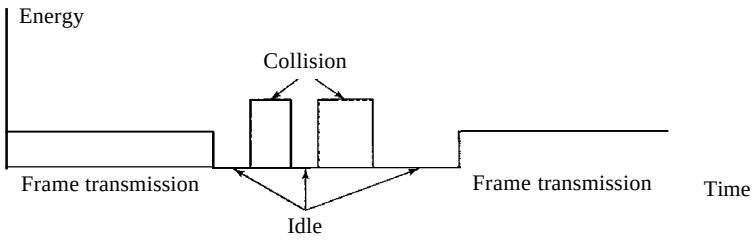
We can say that the level of energy in a channel can have three values: zero, normal, and abnormal. At the zero level, the channel is idle. At the normal level, a station has

Figure 12.14 *Flow diagramfor the CSMAlCD*



successfully captured the channel and is sending its frame. At the abnormal level, there is a collision and the level of the energy is twice the normal level. A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode. Figure 12.15 shows the situation.

Figure 12.15 *Energy level during transmission, idleness, or collision*



Throughput

The throughput of *CSMAlCD* is greater than that of pure or slotted ALOHA. The maximum throughput occurs at a different value of G and is based on the persistence method

and the value of p in the p-persistent approach. For I-persistent method the maximum throughput is around 50 percent when $G = 1$. For nonpersistent method, the maximum throughput can go up to 90 percent when G is between 3 and 8.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

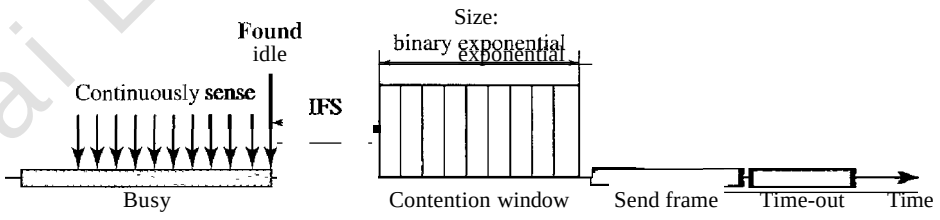
The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles.

However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection.

We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance *(CSMAICA)* was invented for this network. Collisions are avoided through the use of CSMAICA's three strategies: the interframe space, the contention window, and acknowledgments, as shown in Figure 12.16.

Figure 12.16 *Timing in CSMAICA*



Interframe Space (IFS)

First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the interframe space or IFS. Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time (described next). The IFS variable can also be used to prioritize

stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

In CSMAlCA, the IFS can also be used to define the priority of a station or a frame.

Contention Window

The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back-off strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the p-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

In CSMAlCA, if the station finds the channel busy, it does not restart the timer of the contention window; it stops the timer and restarts it when the channel becomes idle.

Acknowledgment

With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

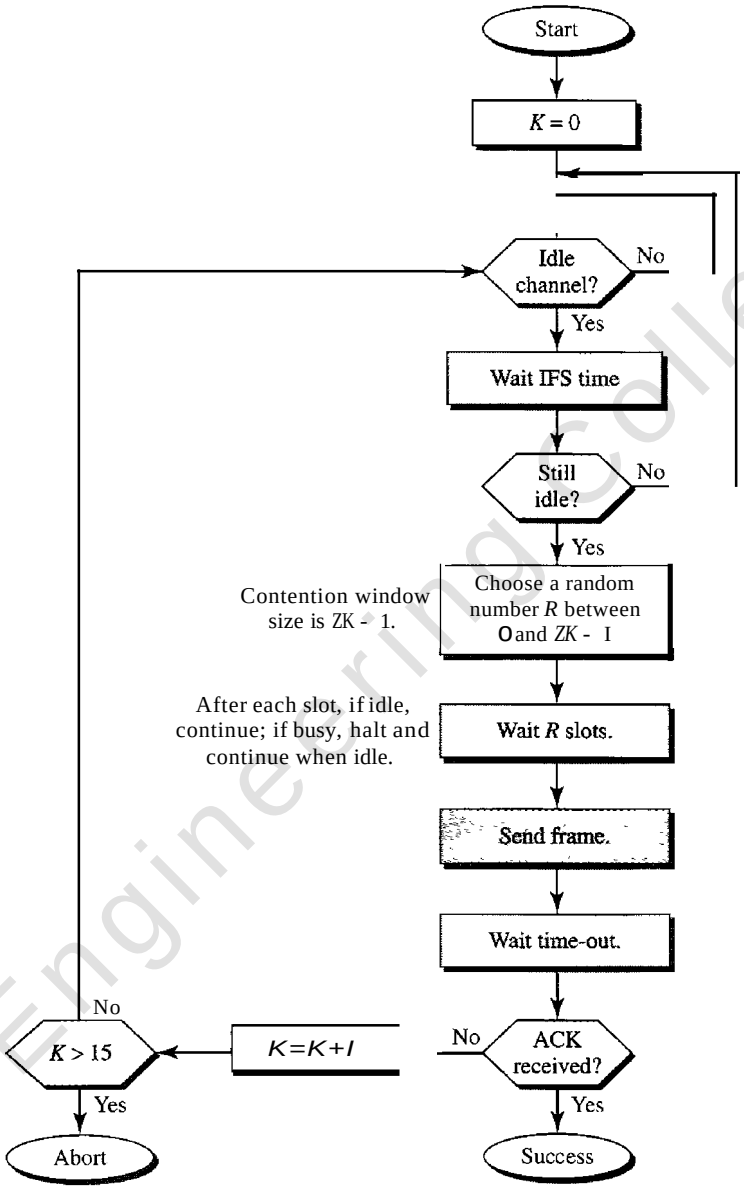
Procedure

Figure 12.17 shows the procedure. Note that the channel needs to be sensed before and after the IFS. The channel also needs to be sensed during the contention time. For each time slot of the contention window, the channel is sensed. If it is found idle, the timer continues; if the channel is found busy, the timer is stopped and continues after the timer becomes idle again.

CSMAICA and Wireless Networks

CSMAICA was mostly intended for use in wireless networks. The procedure described above, however, is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals. We will see how these issues are solved by augmenting the above protocol with hand-shaking features. The use of CSMNCA in wireless networks will be discussed in Chapter 14.

Figure 12.17 *Flow diagramfor CSMAICA*



12.2 CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three popular controlled-access methods.

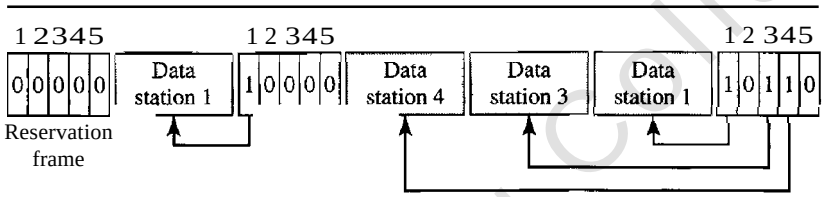
Reservation

In the reservation method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.

Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

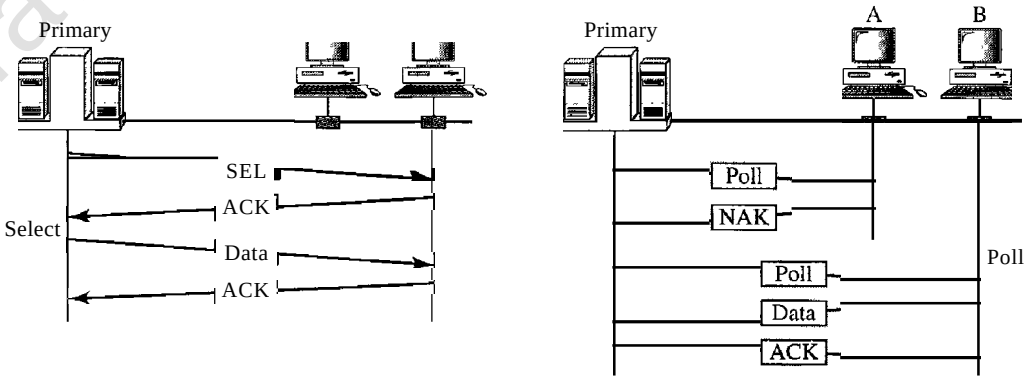
Figure 12.18 *Reservation access method*



Polling

Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations. All data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session (see Figure 12.19).

Figure 12.19 *Select and poll functions in polling access method*



If the primary wants to receive data, it asks the secondaries if they have anything to send; this is called poll function. If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.

Select

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available.

If it has something to send, the primary device sends it. What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

Token Passing

In the token-passing method, the stations in a network are organized in a logical ring. In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

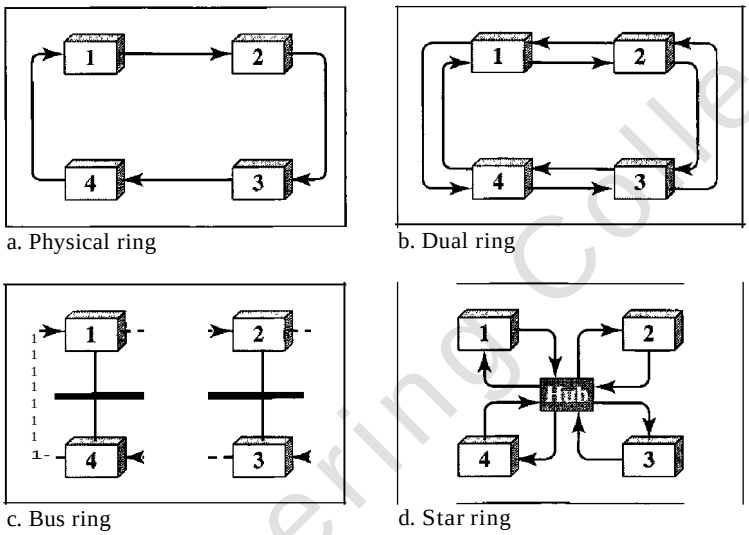
But how is the right to access the channel passed from one station to another? In this method, a special packet called a token circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station.

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed. For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure 12.20 show four different physical topologies that can create a logical ring.

Figure 12.20 *Logical ring and physical topology in token-passing access method*



In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links-the medium between two adjacent stations-fails, the whole system fails.

The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called FDDI (Fiber Distributed Data Interface) and CDDI (Copper Distributed Data Interface) use this topology.

In the bus ring topology, also called a token bus, the stations are connected to a single cable called a bus. They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

In a star ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network

less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

12.3 CHANNELIZATION

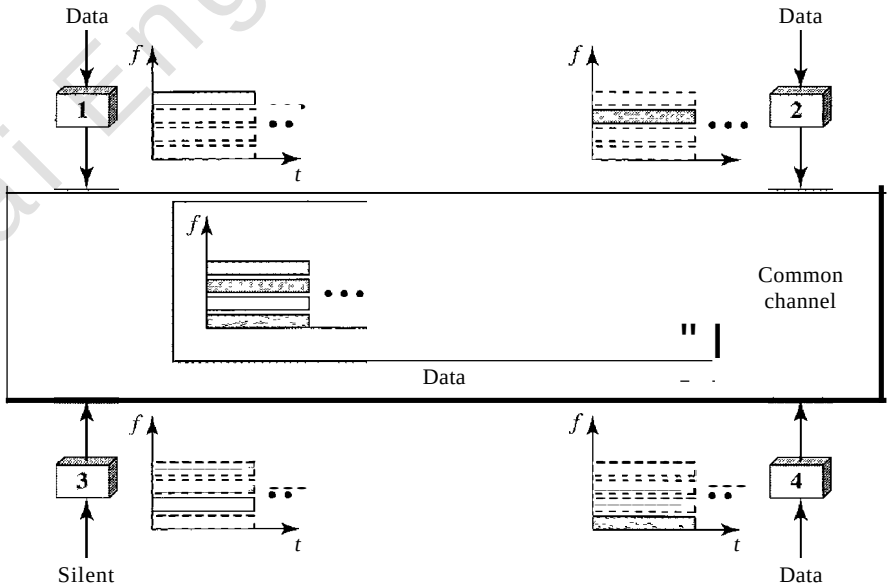
Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

We see the application of all these methods in Chapter 16 when we discuss cellular phone systems.

Frequency-Division Multiple Access (FDMA)

In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a bandpass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small *guard bands.* Figure 12.21 shows the idea of FDMA.

Figure 12.21 *Frequency-division multiple access (FDMA)*



In FDMA, the available bandwidth of the common channel is divided into bands that are separated by guard bands.

FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA. We will see in Chapter 16 how this feature can be used in cellular telephone systems.

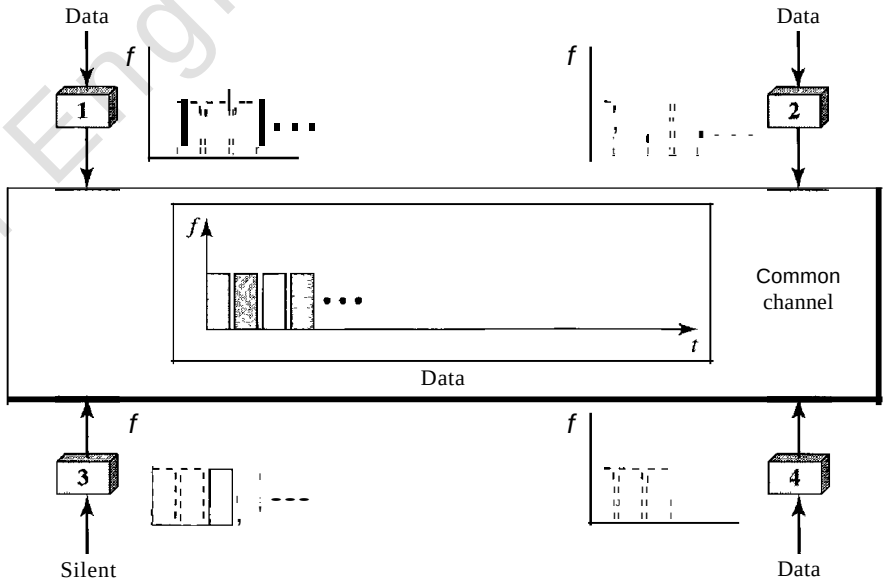
We need to emphasize that although FDMA and FDM conceptually seem similar, there are differences between them. FDM, as we saw in Chapter 6, is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

Time-Division Multiple Access (TDMA)

In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in is assigned time slot. Figure 12.22 shows the idea behind TDMA.

Figure 12.22 *Time-division multiple access (TDMA)*



The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area. To compensate for the delays, we can insert *guard*

times. Synchronization is normally accomplished by having some synchronization bits (normally refened to as preamble bits) at the beginning of each slot.

In TDMA, the bandwidth is just one channel that is timeshared between different stations.

We also need to emphasize that although TDMA and TDM conceptually seem the same, there are differences between them. TDM, as we saw in Chapter 6, is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

TDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

Code-Division Multiple Access (CDMA)

Code-division multiple access (CDMA) was conceived several decades ago. Recent advances in electronic technology have finally made its implementation possible. CDMA differs from FDMA because only one channel occupies the entire bandwidth of the link. It differs from TDMA because all stations can send data simultaneously; there is no timesharing.

In CDMA, one channel carries all transmissions simultaneously.

Analogy

Let us first give an analogy. CDMA simply means communication with different codes. For example, in a large room with many people, two people can talk in English if nobody else understands English. Another two people can talk in Chinese if they are the only ones who understand Chinese, and so on. In other words, the common channel, the space of the room in this case, can easily allow communication between several couples, but in different languages (codes).

Idea

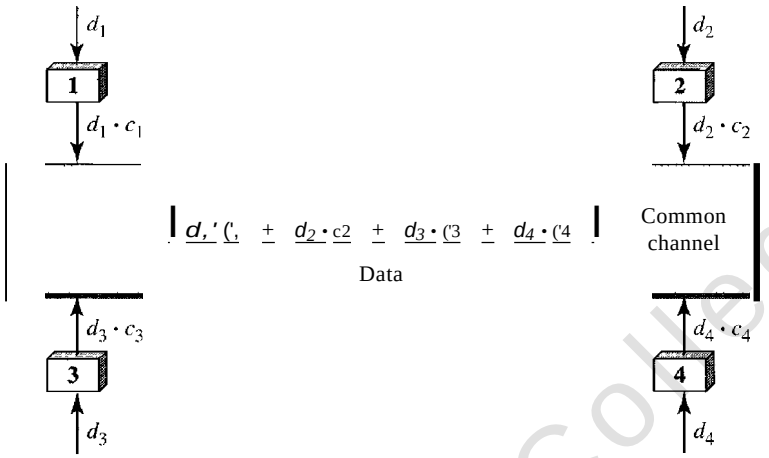
Let us assume we have four stations 1, 2, 3, and 4 connected to the same channel. The data from station 1 are d_1, from station 2 are d_2, and so on. The code assigned to the first station is cI, to the second is $c2$, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure 12.23.

Station 1 mUltiplies (a special kind of multiplication, as we will see) its data by its code to get $d_1 \cdot CI'$ Station 2 multiplies its data by its code to get $d_2 \cdot c2'$ And so on. The

Figure 12.23 *Simple idea of communication with code*



data that go on the channel are the sum of all these terms, as shown in the box. Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station I is saying. It multiplies the data on the channel by cl' the code of station 1.

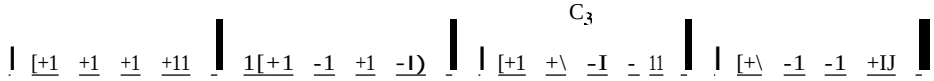
Because (cl . cl) is 4, but *(c2* . cI), *(c3* . cI), and *(c4* . cl) are all Os, station 2 divides the result by 4 to get the data from station 1.

$$\text{data} = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1$$
$$= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1$$

Chips

CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips, as shown in Figure 12.24. The codes are for the previous example.

Figure 12.24 *Chip sequences*



Later in this chapter we show how we chose these sequences. For now, we need to know that we did not choose the sequences randomly; they were carefully selected. They are called orthogonal sequences and have the following properties:

1. Each sequence is made of *N* elements, where *N* is the number of stations.

2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

$$2 \cdot [+1 +1 -1 -1] = [+2 +2 -2 -2]$$

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in the each sequence. This is called the inner product of two equal sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 -1 -1] = 1 + 1 + 1 + 1 = 4$$

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called inner product of two different sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 +1 +1] = 1 + 1 - 1 - 1 = 0$$

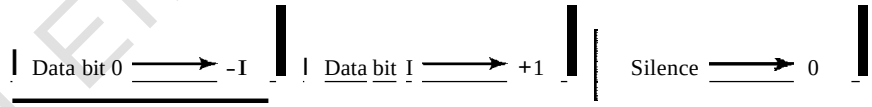
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1 +1 -1 -1] + [+1 +1 +1 +1] = [+2 +2 0 0]$$

Data Representation

We follow these rules for encoding: If a station needs to send a 0 bit, it encodes it as -1; if it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0. These are shown in Figure 12.25.

Figure 12.25 *Data representation in CDMA*



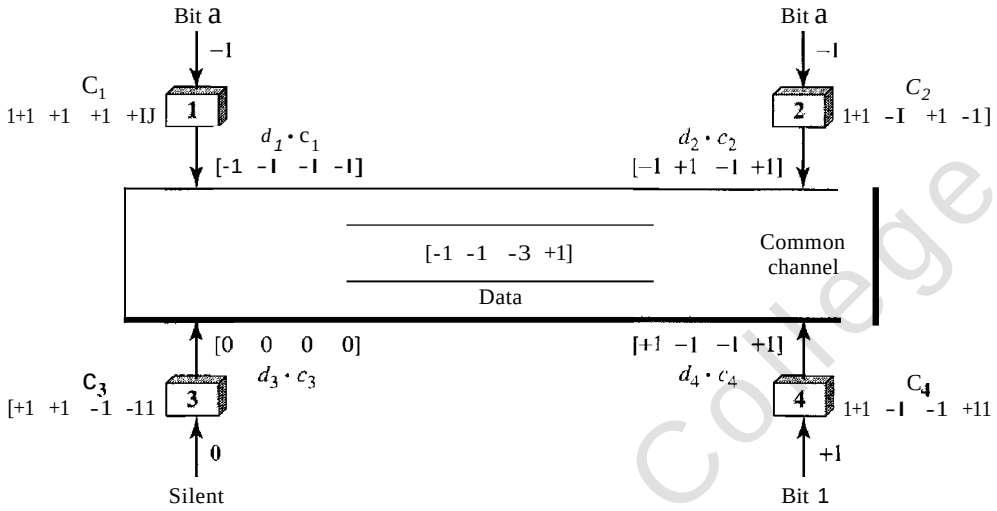
Encoding and Decoding

As a simple example, we show how four stations share the link during a I-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent. The data at the sender site are translated to -1, -1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. Figure 12.26 shows the situation.

Now imagine station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 -1 +1 -1], to get

$$[-1 -1 -3 +1] \cdot [+1 -1 +1 -1] = -4/4 = -1 \Rightarrow \text{bit } 1$$

Figure 12.26 *Sharing channel in CDMA*



Signal Level

The process can be better understood if we show the digital signal produced by each station and the data recovered at the destination (see Figure 12.27). The figure shows the corresponding signals for each station (using NRZ-L for simplicity) and the signal that is on the common channel.

Figure 12.27 *Digital signal created by four stations in CDMA*

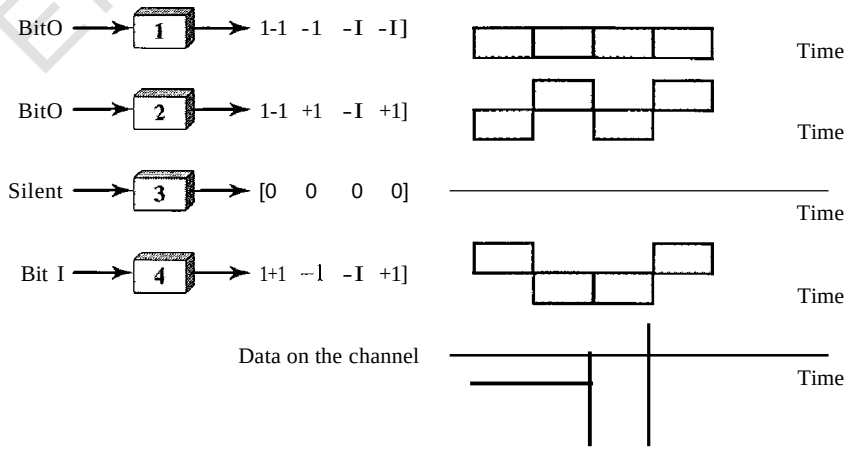
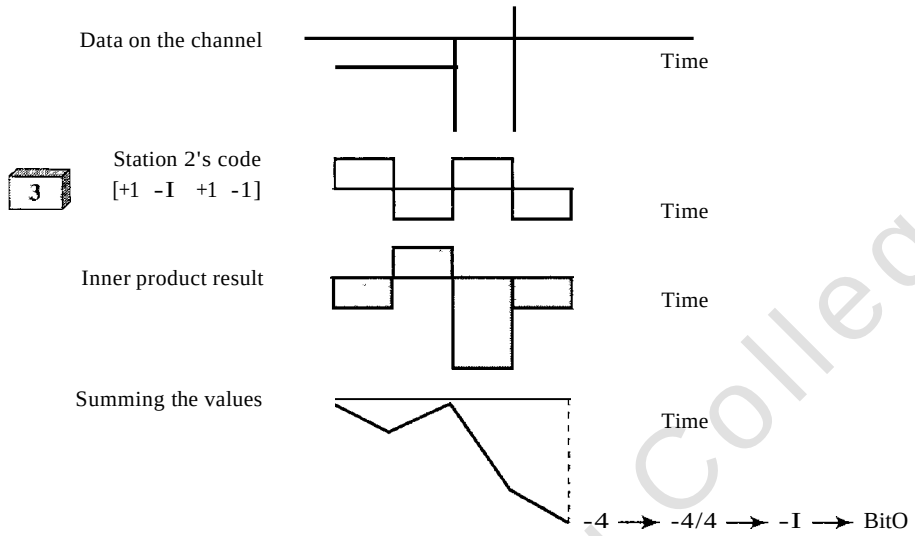


Figure 12.28 shows how station 3 can detect the data sent by station 2 by using the code for station 2. The total data on the channel are multiplied (inner product operation) by the signal representing station 2 chip code to get a new signal. The station then integrates and adds the area under the signal, to get the value -4, which is divided by 4 and interpreted as bit 0.

Figure 12.28 *Decoding of the composite signal for one in CDMA*



Sequence Generation

To generate chip sequences, we use a **Walsh table,** which is a two-dimensional table with an equal number of rows and columns, as shown in Figure 12.29.

Figure 12.29 *General rule and examples of creating Walsh tables*

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \qquad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

a. Two basic rules

$$WI = \begin{bmatrix} +1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \qquad W_4 = \begin{vmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{vmatrix}$$

b. Generation of WI' W_2, and W_4

In the Walsh table, each row is a sequence of chips. *WI* for a one-chip sequence has one row and one column. We can choose -1 or +1 for the chip for this trivial table (we chose +1). According to Walsh, if we know the table for *N* sequences W_N, we can create the table for *2N* sequences W_{2N}, as shown in Figure 12.29. The W_N with the overbar $\overline{W_N}$ stands for the complement of W_N, where each +1 is changed to -1 and vice versa. Figure 12.29 also shows how we can create W_2 and W_4 from *WI'* After we select *WI*, W_2

can be made from four W_j's, with the last one the complement of WI' After W_2 is generated, W_4 can be made of four W_2's, with the last one the complement of W_2. Of course, W_S is composed of four W_4's, and so on. Note that after W_N is made, each station is assigned a chip corresponding to a row.

Something we need to emphasize is that the number of sequences N needs to be a power of 2. In other words, we need to have $N = 2^m$.

The number of sequences in a Walsh table needs to be $N = 2^m$.

Example 12.6

Find the chips for a network with

a. Two stations
b. Four stations

Solution

We can use the rows of W_2 and W_4 in Figure 12.29:

a. For a two-station network, we have [+1 +1] and [+1 -1].
b. For a four-station network we have [+1 +1 +1 +1], [+1 -1 +1 -1], [+1 +1 -1 −1], and [+1 -1 -1 +1].

Example 12.7

What is the number of sequences if we have 90 stations in our network?

Solution

The number of sequences needs to be 2^m. We need to choose $m = 7$ and $N = 2^7$ or 128. We can then use 90 of the sequences as the chips.

Example 12.8

Prove that a receiving station can get the data sent by a specific sender if it multiplies the entire data on the channel by the sender's chip code and then divides it by the number of stations.

Solution

Let us prove this for the first station, using our previous four-station example. We can say that the data on the channel $D = Cd_1 . ("1 + d_2 . ("2 + d_3 . ("3 + d_4 . ("4)'$ The receiver which wants to get the data sent by station I multiplies these data by $("I'$

$$
\begin{aligned}
D . ("1 &= (d_1 \cdot ("1 + d_2 . ("2 + d_3 . ("3 + d_4 . ("4) . ("1 \\
&= d_1 . ("1 . ("1 + d_2 . ("2 . ("1 + d_3 . ("3 . ("1 + d_4 . ("4 . ("1 \\
&= d_I \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\
&= d_I \times N
\end{aligned}
$$

When we divide the result by N, we get d_I'

12.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Multiple access is discussed in Chapter 4 of [Tan03], Chapter 16 of [Sta04], Chapter 6 of [GW04], and Chapter 8 of [For03]. More advanced materials can be found in [KMK04].

12.5 KEY TERMS

I-persistent method
ALOHA
binary exponential backup
carrier sense multiple access (CSMA)
carrier sense multiple access with collision avoidance *(CSMA/CA)*
carrier sense multiple access with collision detection *(CSMA/CD)*
channelization
code-division multiple access (CDMA)
collision
contention
controlled access
frequency-division multiple access (FDMA)
inner product
interframe space (IFS)
jamming signal
multiple access (MA)
nonpersistent method
orthogonal sequence
polling
p-persistent method
primary station
propagation time
pure ALOHA
random access
reservation
secondary station
slotted ALOHA
time-division multiple access (TDMA)
token
token passing
vulnerable time
Walsh table

12.6 SUMMARY

- We can consider the data link layer as two sublayers. The upper sublayer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media.
- Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups: random access protocols, controlled access protocols, and channelization protocols.
- In random access or contention methods, no station is superior to another station and none is assigned the control over another.
- ALOHA allows multiple access (MA) to the shared medium.There are potential collisions in this arrangement. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.
- To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station

senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium before sending. Three methods have been devised for carrier sensing: I-persistent, nonpersistent, and p-persistent.

- Carrier sense multiple access with collision detection (CSMA/CD) augments the CSMA algorithm to handle collision. In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.
- To avoid collisions on wireless networks, carrier sense multiple access with collision avoidance (CSMA/CA) was invented. Collisions are avoided through the use three strategies: the interframe space, the contention window, and acknowledgments.
- In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discussed three popular controlled-access methods: reservation, polling, and token passing.
- In the reservation access method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.
- In the polling method, all data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions.
- In the token-passing method, the stations in a network are organized in a logical ring. Each station has a predecessor and a successor. A special packet called a token circulates through the ring.
- Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. We discussed three channelization protocols: FDMA, TDMA, and CDMA.
- In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time.
- In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot.
- In code-division multiple access (CDMA), the stations use different codes to achieve multiple access. CDMA is based on coding theory and uses sequences of numbers called chips. The sequences are generated using orthogonal codes such the Walsh tables.

12.7 PRACTICE SET

Review Questions

1. List three categories of multiple access protocols discussed in this chapter.
2. Define random access and list three protocols in this category.

3. Define controlled access and list three protocols in this category.
4. Define channelization and list three protocols in this category.
5. Explain why collision is an issue in a random access protocol but not in controlled access or channelizing protocols.
6. Compare and contrast a random access protocol with a controlled access protocol.
7. Compare and contrast a random access protocol with a channelizing protocol.
8. Compare and contrast a controlled access protocol with a channelizing protocol.
9. Do we need a multiple access protocol when we use the 1ocalloop of the telephone company to access the Internet? Why?
10. Do we need a multiple access protocol when we use one CATV channel to access the Internet? Why?

Exercises

11. We have a pure ALOHA network with 100 stations. If $T_{fr} = 1\,\mu s$, what is the number of frames/s each station can send to achieve the maximum efficiency.
12. Repeat Exercise 11 for slotted ALOHA.
13. One hundred stations on a pure ALOHA network share a l-Mbps channel. If frames are 1000 bits long, find the throughput if each station is sending 10 frames per second.
14. Repeat Exercise 13 for slotted ALOHA.
15. In a *CDMAICD* network with a data rate of 10 Mbps, the minimum frame size is found to be 512 bits for the correct operation of the collision detection process. What should be the minimum frame size if we increase the data rate to 100 Mbps? To 1 Gbps? To 10 Gbps?
16. In a *CDMAICD* network with a data rate of 10 Mbps, the maximum distance between any station pair is found to be 2500 m for the correct operation of the collision detection process. What should be the maximum distance if we increase the data rate to 100 Mbps? To 1 Gbps? To 10 Gbps?
17. In Figure 12.12, the data rate is 10 Mbps, the distance between station A and C is 2000 m, and the propagation speed is 2×10^8 *mls.* Station A starts sending a long frame at time $t1 = 0$; station C starts sending a long frame at time $t2 = 3\,\mu s$. The size of the frame is long enough to guarantee the detection of collision by both stations. Find:
    a. The time when station C hears the collision *(t3)'*
    b. The time when station A hears the collision *(t4)'*
    c. The number of bits station A has sent before detecting the collision.
    d. The number of bits station C has sent before detecting the collision.
18. Repeat Exercise 17 if the data rate is 100 Mbps.
19. Calculate the Walsh table W_S from W_4 in Figure 12.29.
20. Recreate the W_2 and W_4 tables in Figure 12.29 using $WI = [-1]$. Compare the recreated tables with the ones in Figure 12.29.
21. Prove the third and fourth orthogonal properties of Walsh chips for W_4 in Figure 12.29.

22. Prove the third and fourth orthogonal properties of Walsh chips for W_4 recreated in Exercise 20.
23. Repeat the scenario depicted in Figures 12.27 to 12.28 if both stations 1 and 3 are silent.
24. A network with one primary and four secondary stations uses polling. The size of a data frame is 1000 bytes. The size of the poll, ACK, and NAK frames are 32 bytes each. Each station has 5 frames to send. How many total bytes are exchanged if there is no limitation on the number of frames a station can send in response to a poll?
25. Repeat Exercise 24 if each station can send only one frame in response to a poll.

Research Activities

26. Can you explain why the vulnerable time in ALOHA depends on T_{Fr}, but in CSMA depends on T_p?
27. In analyzing ALOHA, we use only one parameter, time; in analyzing CSMA, we use two parameters, space and time. Can you explain the reason?

CHAPTER 13

Wired LANs: Ethernet

In Chapter 1, we learned that a local area network (LAN) is a computer network that is designed for a limited geographic area such as a building or a campus. Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet.

The LAN market has seen several technologies such as Ethernet, Token Ring, Token Bus, FDDI, and ATM LAN. Some of these technologies survived for a while, but Ethernet is by far the dominant technology.

In this chapter, we first briefly discuss the IEEE Standard Project 802, designed to regulate the manufacturing and interconnectivity between different LANs. We then concentrate on the Ethernet LANs.

Although Ethernet has gone through a four-generation evolution during the last few decades, the main concept has remained. Ethernet has changed to meet the market needs and to make use of the new technologies.

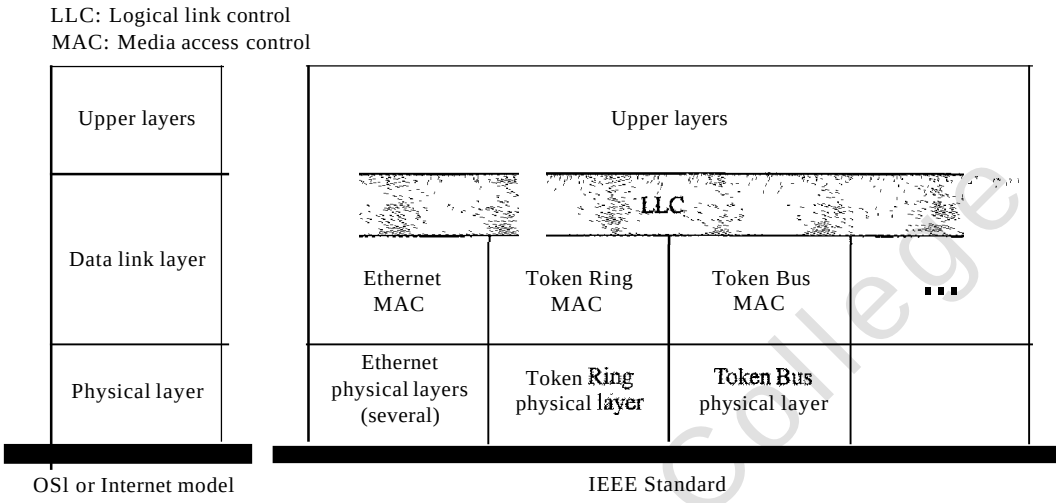
13.1 IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 does not seek to replace any part of the OSI or the Internet model. Instead, it is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

The standard was adopted by the American National Standards Institute (ANSI). In 1987, the International Organization for Standardization (ISO) also approved it as an international standard under the designation ISO 8802.

The relationship of the 802 Standard to the traditional OSI model is shown in Figure 13.1. The IEEE has subdivided the data link layer into two sublayers: logical link control (LLC) and media access control (MAC). IEEE has also created several physicallayer standards for different LAN protocols.

Figure 13.1 *IEEE standardfor LANs*



Data Link Layer

As we mentioned before, the data link layer in the IEEE standard is divided into two sublayers: LLC and MAC.

Logical Link Control (LLC)

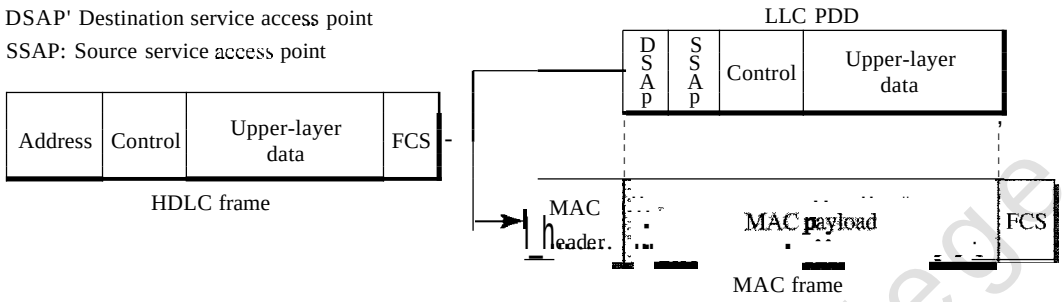
In Chapter II, we discussed data link control. We said that data link control handles framing, flow control, and error control. In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control. Framing is handled in both the LLC sublayer and the MAC sublayer.

The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LANs. A single LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent. Figure 13.1 shows one single LLC protocol serving several MAC protocols.

Framing LLC defines a protocol data unit (PDU) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the upper-layer protocol at the source and destination that uses LLC. These fields are called the destination service access point (DSAP) and the source service access point (SSAP). The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. In other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sublayer, as shown in Figure 13.2.

Need for LLC The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services. For example, if a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols. However, most upper-layer protocols

Figure 13.2 *HDLC frame compared with LLC and MAC frames*



such as IP (discussed in Chapter 20), do not use the services of LLC. For this reason, we end our discussion of LLC.

Media Access Control (MAC)

In Chapter 12, we discussed multiple access methods including random access, controlled access, and channelization. IEEE Project 802 has created a sublayer called media access control that defines the specific access method for each LAN. For example, it defines *CSMA/CD* as the media access method for Ethernet LANs and the token-passing method for Token Ring and Token Bus LANs. As we discussed in the previous section, part of the framing function is also handled by the MAC layer.

In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.

Physical Layer

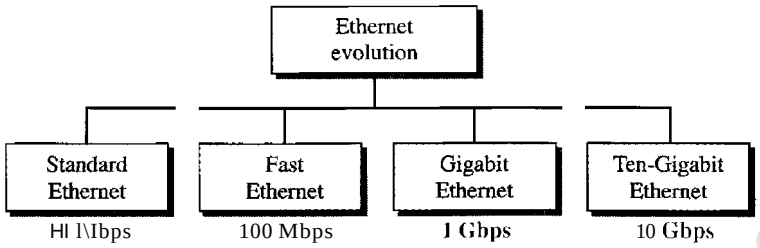
The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specifications for each Ethernet implementations as we will see later.

13.2 STANDARD ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations: Standard Ethernet (1ot Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (l Gbps), and Ten-Gigabit Ethernet (l0 Gbps), as shown in Figure 13.3. We briefly discuss all these generations starting with the first, Standard (or traditional) Ethernet.

t Ethernet defined some I-Mbps protocols, but they did not survive.

Figure 13.3 *Ethernet evolution through four generations*



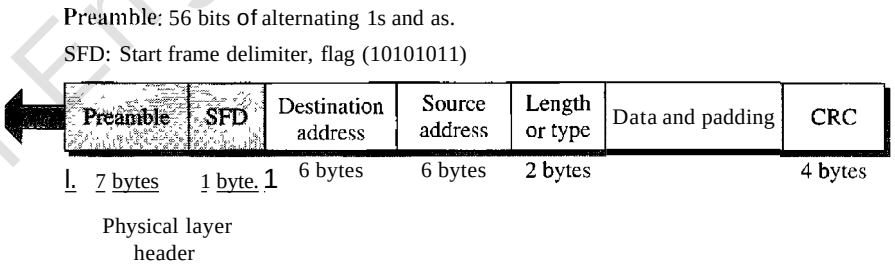
MAC Sublayer

In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRe. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers. The format ofthe MAC frame is shown in Figure 13.4.

Figure 13.4 *802.3 MAC frame*



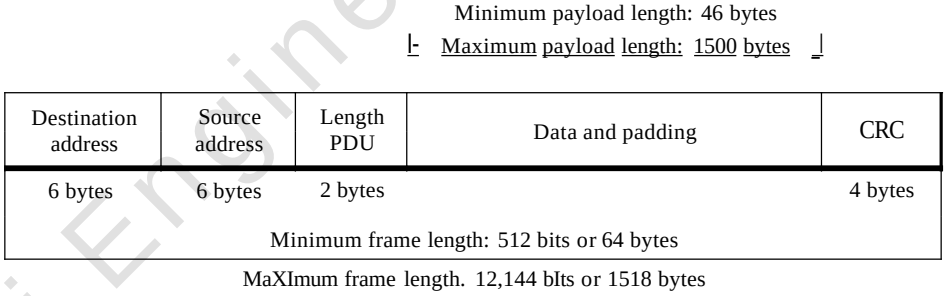
- Preamble. The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and Is that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The preamble is actually added at the physical layer and is not (formally) part of the frame.
- Start frame delimiter (SFD). The second field (I byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

- Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet. We will discuss addressing shortly.
- Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet. We will discuss addressing shortly.
- Length or type. This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.
- Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes, as we will see later.
- CRC. The last field contains error detection information, in this case a CRC-32 (see Chapter 10).

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in Figure 13.5.

Figure 13.5 *Minimum and maximum lengths*



The minimum length restriction is required for the correct operation of CSMA/CD as we will see shortly. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 - 18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference.

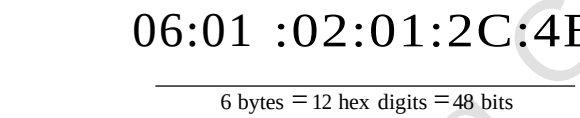
The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Frame length:
Minimum: 64 bytes (512 bits) Maximum: 1518 bytes (12,144 bits)

Addressing

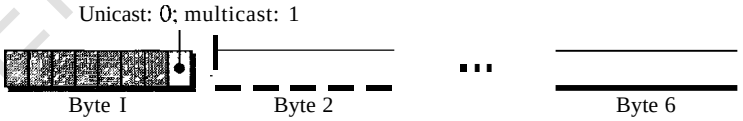
Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a 6-byte physical address. As shown in Figure 13.6, the Ethernet address is 6 bytes (48 bits), nonnally written in hexadecimal notation, with a colon between the bytes.

Figure 13.6 *Example ofan Ethernet address in hexadecimal notation*



Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address-the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. Figure 13.7 shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.

Figure 13.7 *Unicast and multicast addresses*



The least significant bit of the first byte defines the type of address.
If the bit is 0, the address is unicast; otherwise, it is multicast.

A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one-to-one. A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one-to-many.

The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty-eight Is.

The broadcast destination address is a special case of
the multicast address in which all bits are Is.

Example 13.1

Define the type of the following destination addresses:

a. 4A:30:10:21:1O:1A
b. 47:20:1B:2E:08:EE
c. FF:FF:FF:FF:FF:FF

Solution

To find the type of the address, we need to look at the second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are F's, the address is broadcast. Therefore, we have the following:

a. This is a unicast address because A in binary is 1010 (even).
b. This is a multicast address because 7 in binary is 0111 (odd).
c. This is a broadcast address because all digits are F's.

The way the addresses are sent out on line is different from the way they are written in hexadecimal notation. The transmission is left-to-right, byte by byte; however, for each byte, the least significant bit is sent first and the most significant bit is sent last. This means that the bit that defines an address as unicast or multicast arrives first at the receIver.

Example 13.2

Show how the address 47:20:1B:2E:08:EE is sent out on line.

Solution

The address is sent left-to-right, byte by byte; for each byte, it is sent right-to-Ieft, bit by bit, as shown below:

11100010 00000100 11011000 01110100 00010000 01110111

Access Method: CSMAICD

Standard Ethernet uses I-persistent CSMAlCD (see Chapter 12).

Slot Time In an Ethernet network, the round-trip time required for a frame to travel from one end of a maximum-length network to the other plus the time needed to send the jam sequence is called the slot time.

$$\text{Slot time} = \text{round-trip time} + \text{time required to send the jam sequence}$$

The slot time in Ethernet is defined in bits. It is the time required for a station to send 512 bits. This means that the actual slot time depends on the data rate; for traditional 10-Mbps Ethernet it is 51.2 μs.

Slot Time and Collision The choice of a 512-bit slot time was not accidental. It was chosen to allow the proper functioning of CSMAlCD. To understand the situation, let us consider two cases.

In the first case, we assume that the sender sends a minimum-size packet of 512 bits. Before the sender can send the entire packet out, the signal travels through the network

and reaches the end of the network. If there is another signal at the end of the network (worst case), a collision occurs. The sender has the opportunity to abort the sending of the frame and to send a jam sequence to inform other stations of the collision. The round-trip time plus the time required to send the jam sequence should be less than the time needed for the sender to send the minimum frame, 512 bits. The sender needs to be aware of the collision before it is too late, that is, before it has sent the entire frame.

In the second case, the sender sends a frame larger than the minimum size (between 512 and 1518 bits). In this case, if the station has sent out the first 512 bits and has not heard a collision, it is guaranteed that collision will never occur during the transmission of this frame. The reason is that the signal will reach the end of the network in less than one-half the slot time. If all stations follow the *CSMA/CD* protocol, they have already sensed the existence of the signal (carrier) on the line and have refrained from sending. If they sent a signal on the line before one-half of the slot time expired, a collision has occurred and the sender has sensed the collision. In other words, collision can only occur during the first half of the slot time, and if it does, it can be sensed by the sender during the slot time. This means that after the sender sends the first 512 bits, it is guaranteed that collision will not occur during the transmission of this frame. The medium belongs to the sender, and no other station will use it. In other words, the sender needs to listen for a collision only during the time the first 512 bits are sent.

Of course, all these assumptions are invalid if a station does not follow the *CSMAlCD* protocol. In this case, we do not have a collision, we have a corrupted station.

Slot Time and Maximum Network Length There is a relationship between the slot time and the maximum length of the network (collision domain). It is dependent on the propagation speed of the signal in the particular medium. In most transmission media, the signal propagates at 2×10^8 rnls (two-thirds of the rate for propagation in air). For traditional Ethernet, we calculate

$$\text{MaxLength} = \text{PropagationSpeed} \times \frac{\text{SlotTime}}{2}$$

$$\text{MaxLength} = (2\text{x to}^8) \text{ X}(5\text{L2 X}10^{-6}(2) = 5120\text{m}$$

Of course, we need to consider the delay times in repeaters and interfaces, and the time required to send the jam sequence. These reduce the maximum-length of a traditional Ethernet network to 2500 m, just 48 percent of the theoretical calculation.

$$\text{MaxLength} = 2500 \text{ m}$$

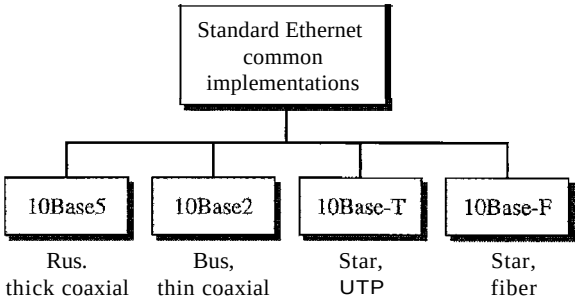
Physical Layer

The Standard Ethernet defines several physical layer implementations; four of the most common, are shown in Figure 13.8.

Encoding and Decoding

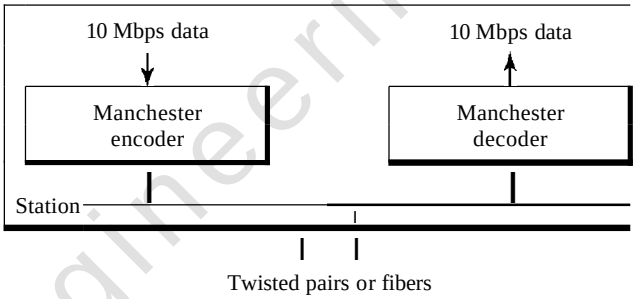
All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the

Figure 13.8 *Categories of Standard Ethernet*



received signal is interpreted as Manchester and decoded into data. As we saw in Chapter 4, Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure 13.9 shows the encoding scheme for Standard Ethernet.

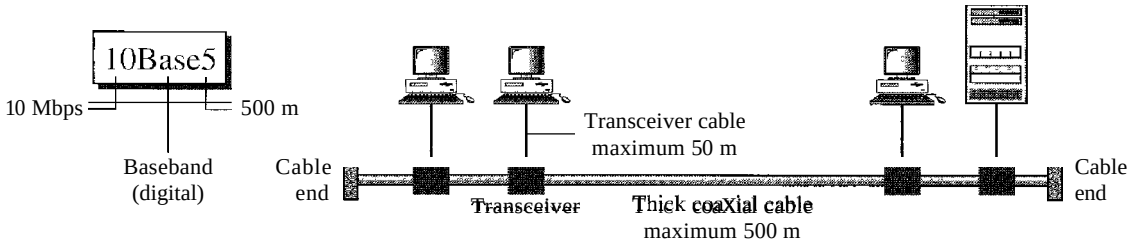
Figure 13.9 *Encoding in a Standard Ethernet implementation*



lOBase5: Thick Ethernet

The first implementation is called **10BaseS, thick Ethernet, or Thicknet.** The nickname derives from the size of the cable, which is roughly the size of a garden hose and too stiff to bend with your hands. lOBaseS was the first Ethernet specification to use a bus topology with an external **transceiver** (transmitter/receiver) connected via a tap to a thick coaxial cable. Figure 13.10 shows a schematic diagram of a lOBase5 implementation.

Figure 13.10 *IOBase5 implementation*



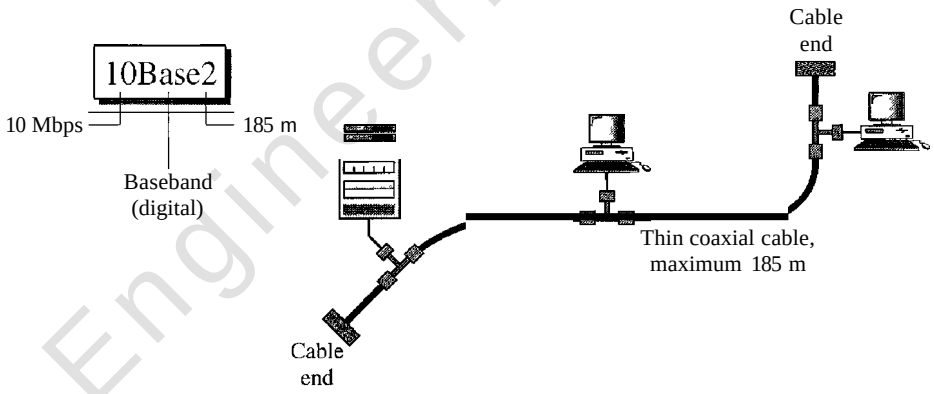
The transceiver is responsible for transmitting, receiving, and detecting collisions. The transceiver is connected to the station via a transceiver cable that provides separate paths for sending and receiving. This means that collision can only happen in the coaxial cable.

The maximum length of the coaxial cable must not exceed *500* m, otherwise, there is excessive degradation of the signal. If a length of more than *500* m is needed, up to five segments, each a maximum of SOO-meter, can be connected using repeaters. Repeaters will be discussed in Chapter 15.

10Base2: Thin Ethernet

The second implementation is called lOBase2, **thin** Ethernet, or Cheapernet. IOBase2 also uses a bus topology, but the cable is much thinner and more flexible. The cable can be bent to pass very close to the stations. In this case, the transceiver is normally part of the network interface card (NIC), which is installed inside the station. Figure 13.11 shows the schematic diagram of a IOBase2 implementation.

Figure 13.11 *lOBase2 implementation*



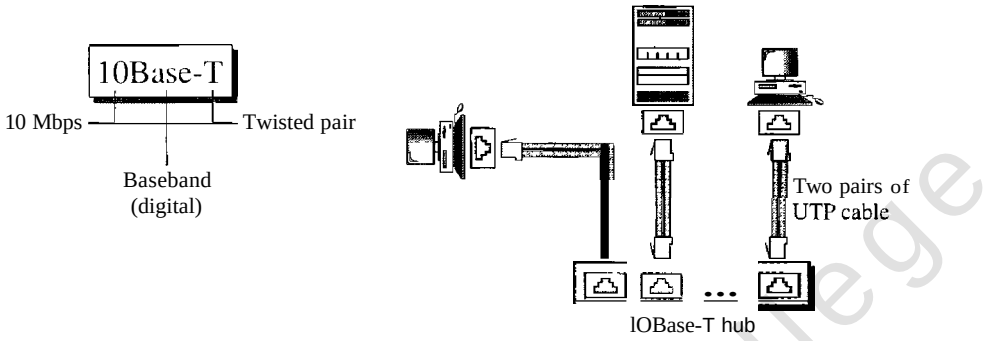
Note that the collision here occurs in the thin coaxial cable. This implementation is more cost effective than 10BaseS because thin coaxial cable is less expensive than thick coaxial and the tee connections are much cheaper than taps. Installation is simpler because the thin coaxial cable is very flexible. However, the length of each segment cannot exceed *185* m (close to 200 m) due to the high level of attenuation in thin coaxial cable.

lOBase-T: Twisted-Pair Ethernet

The third implementation is called lOBase-T or twisted-pair Ethernet. 1OBase-T uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable, as shown in Figure 13.12.

Note that two pairs of twisted cable create two paths (one for sending and one for receiving) between the station and the hub. Any collision here happens in the hub. Compared to lOBaseS or lOBase2, we can see that the hub actually replaces the coaxial

Figure 13.12 *IOBase-T implementation*

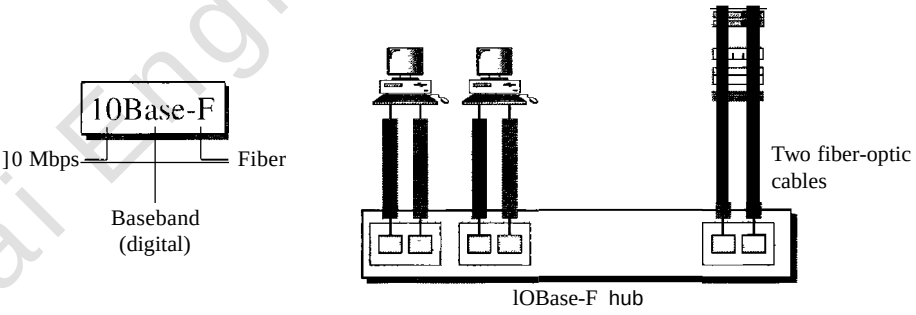


cable as far as a collision is concerned. The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable.

lOBase-F: Fiber Ethernet

Although there are several types of optical fiber lO-Mbps Ethernet, the most common is called 10Base-F. lOBase-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables, as shown in Figure 13.13.

Figure 13.13 *IOBase-F implementation*



Summary

Table 13.1 shows a summary of Standard Ethernet implementations.

Table 13.1 *Summary of Standard Ethernet implementations*

Characteristics	*lOBase5*	*lOBase2*	*lOBase-T*	*IOBase-F*
Media	Thick coaxial cable	Thin coaxial cable	2UTP	2 Fiber
Maximum length	500m	185 m	100m	2000m
Line encoding	Manchester	Manchester	Manchester	Manchester

13.3 CHANGES IN THE STANDARD

The 10-Mbps Standard Ethernet has gone through several changes before moving to the higher data rates. These changes actually opened the road to the evolution of the Ethernet to become compatible with other high-data-rate LANs. We discuss some of these changes in this section.

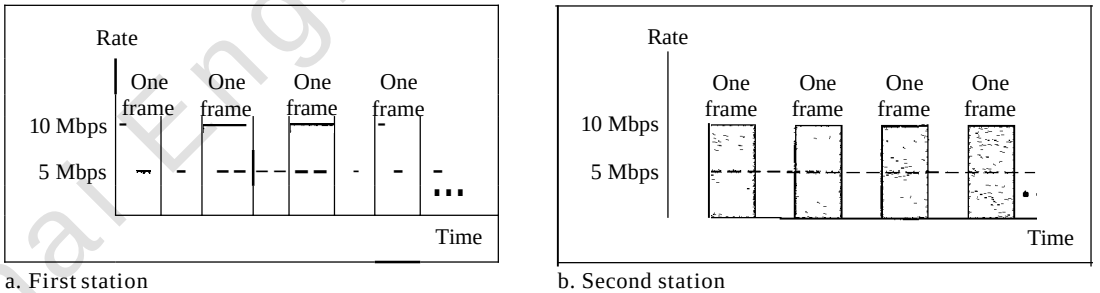
Bridged Ethernet

The first step in the Ethernet evolution was the division of a LAN by bridges. Bridges have two effects on an Ethernet LAN: They raise the bandwidth and they separate collision domains. We discuss bridges in Chapter 15.

Raising the Bandwidth

In an unbridged Ethernet network, the total capacity (10 Mbps) is shared among all stations with a frame to send; the stations share the bandwidth of the network. If only one station has frames to send, it benefits from the total capacity (10 Mbps). But if more than one station needs to use the network, the capacity is shared. For example, if two stations have a lot of frames to send, they probably alternate in usage. When one station is sending, the other one refrains from sending. We can say that, in this case, each station on average, sends at a rate of 5 Mbps. Figure 13.14 shows the situation.

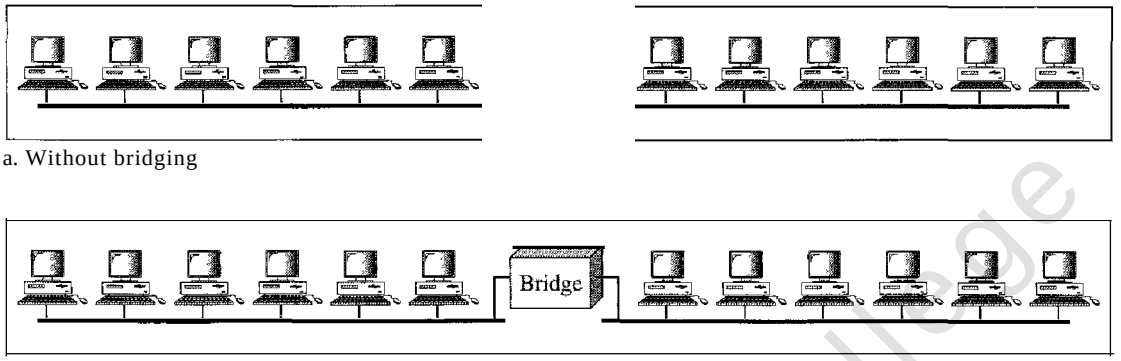
Figure 13.14 *Sharing bandwidth*



The bridge, as we will learn in Chapter 15, can help here. A bridge divides the network into two or more networks. Bandwidth-wise, each network is independent. For example, in Figure 13.15, a network with 12 stations is divided into two networks, each with 6 stations. Now each network has a capacity of 10 Mbps. The lO-Mbps capacity in each segment is now shared between 6 stations (actually 7 because the bridge acts as a station in each segment), not 12 stations. In a network with a heavy load, each station theoretically is offered 10/6 Mbps instead of 10/12 Mbps, assuming that the traffic is not going through the bridge.

It is obvious that if we further divide the network, we can gain more bandwidth for each segment. For example, if we use a four-port bridge, each station is now offered 10/3 Mbps, which is 4 times more than an unbridged network.

Figure 13.15 *A network with and without a bridge*



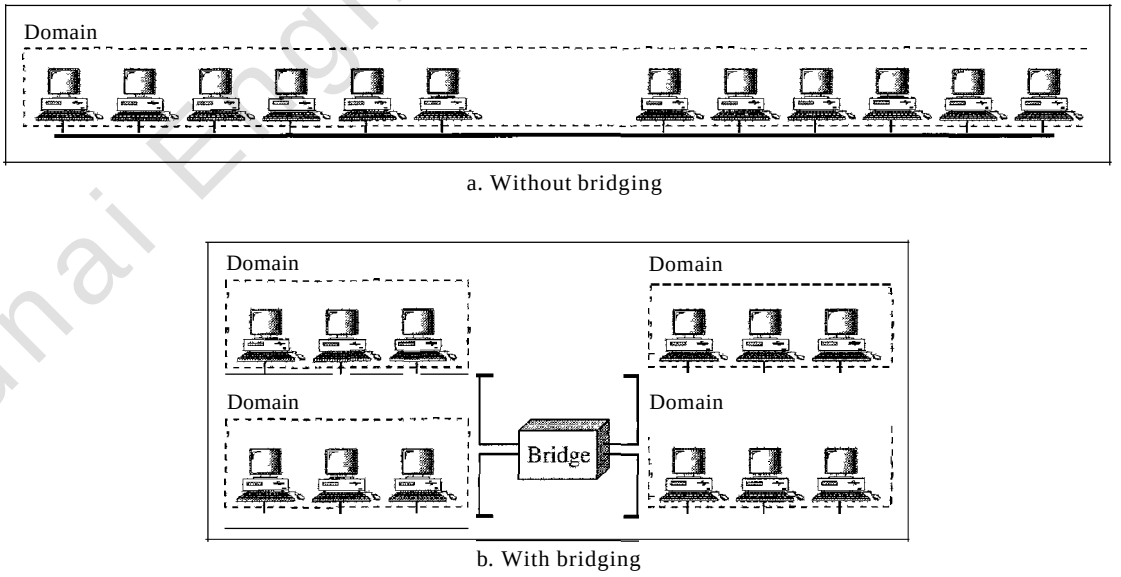
a. Without bridging

b. With bridging

Separating Collision Domains

Another advantage of a bridge is the separation of the collision domain. Figure 13.16 shows the collision domains for an unbridged and a bridged network. You can see that the collision domain becomes much smaller and the probability of collision is reduced tremendously. Without bridging, 12 stations contend for access to the medium; with bridging only 3 stations contend for access to the medium.

Figure 13.16 *Collision domains in an unbridged network and a bridged network*



a. Without bridging

b. With bridging

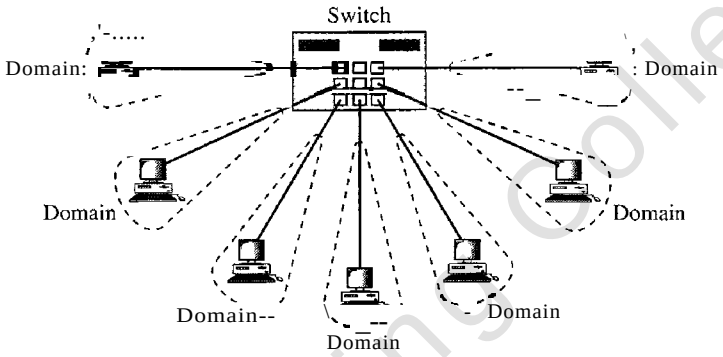
Switched Ethernet

The idea of a bridged LAN can be extended to a switched LAN. Instead of having two to four networks, why not have *N* networks, where *N* is the number of stations on the LAN? In other words, if we can have a multiple-port bridge, why not have an *N-port*

switch? In this way, the bandwidth is shared only between the station and the switch (5 Mbps each). In addition, the collision domain is divided into *N* domains.

A layer 2 switch is an *N-port* bridge with additional sophistication that allows faster handling of the packets. Evolution from a bridged Ethernet to a switched Ethernet was a big step that opened the way to an even faster Ethernet, as we will see. Figure 13.17 shows a switched LAN.

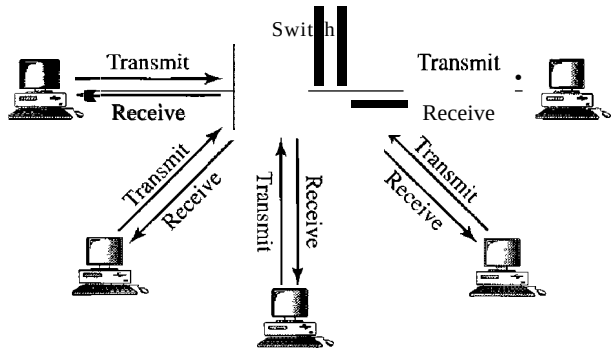
Figure 13.17 *Switched Ethernet*



Full-Duplex Ethernet

One of the limitations of 10Base5 and lOBase2 is that communication is half-duplex (lOBase-T is always full-duplex); a station can either send or receive, but may not do both at the same time. The next step in the evolution was to move from switched Ethernet to full-duplex switched Ethernet. The full-duplex mode increases the capacity of each domain from 10 to 20 Mbps. Figure 13.18 shows a switched Ethernet in full-duplex mode. Note that instead of using one link between the station and the switch, the configuration uses two links: one to transmit and one to receive.

Figure 13.18 *Full-duplex switched Ethernet*



No Needfor CSMAICD

In full-duplex switched Ethernet, there is no need for the *CSMAICD* method. In a full-duplex switched Ethernet, each station is connected to the switch via two separate links.

Each station or switch can send and receive independently without worrying about collision. Each link is a point-to-point dedicated path between the station and the switch. There is no longer a need for carrier sensing; there is no longer a need for collision detection. The job of the MAC layer becomes much easier. The carrier sensing and collision detection functionalities of the MAC sublayer can be turned off.

MAC Control Layer

Standard Ethernet was designed as a connectionless protocol at the MAC sublayer. There is no explicit flow control or error control to inform the sender that the frame has arrived at the destination without error. When the receiver receives the frame, it does not send any positive or negative acknowledgment.

To provide for flow and error control in full-duplex switched Ethernet, a new sublayer, called the MAC control, is added between the LLC sublayer and the MAC sublayer.

13.4 FAST ETHERNET

Fast Ethernet was designed to compete with LAN protocols such as FDDI or Fiber Channel (or Fibre Channel, as it is sometimes spelled). IEEE created Fast Ethernet under the name 802.3u. Fast Ethernet is backward-compatible with Standard Ethernet, but it can transmit data 10 times faster at a rate of 100 Mbps. The goals of Fast Ethernet can be summarized as follows:

1. Upgrade the data rate to 100 Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bit address.
4. Keep the same frame format.
5. Keep the same minimum and maximum frame lengths.

MAC Sublayer

A main consideration in the evolution of Ethernet from 10 to 100 Mbps was to keep the MAC sublayer untouched. However, a decision was made to drop the bus topologies and keep only the star topology. For the star topology, there are two choices, as we saw before: half duplex and full duplex. In the half-duplex approach, the stations are connected via a hub; in the full-duplex approach, the connection is made via a switch with buffers at each port.

The access method is the same *(CSMAlCD)* for the half-duplex approach; for full-duplex Fast Ethernet, there is no need for *CSMAlCD*. However, the implementations keep *CSMA/CD* for backward compatibility with Standard Ethernet.

Autonegotiatioll

A new feature added to Fast Ethernet is called autonegotiation. It allows a station or a hub a range of capabilities. Autonegotiation allows two devices to negotiate the mode

or data rate of operation. It was designed particularly for the following purposes:

- To allow incompatible devices to connect to one another. For example, a device with a maximum capacity of 10 Mbps can communicate with a device with a 100 Mbps capacity (but can work at a lower rate).
- To allow one device to have multiple capabilities.
- To allow a station to check a hub's capabilities.

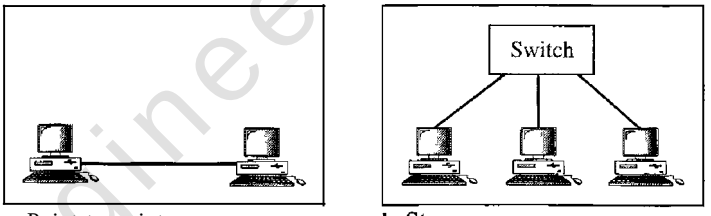
Physical Layer

The physical layer in Fast Ethernet is more complicated than the one in Standard Ethernet. We briefly discuss some features of this layer.

Topology

Fast Ethernet is designed to connect two or more stations together. If there are only two stations, they can be connected point-to-point. Three or more stations need to be connected in a star topology with a hub or a switch at the center, as shown in Figure 13.19.

Figure 13.19 *Fast Ethernet topology*



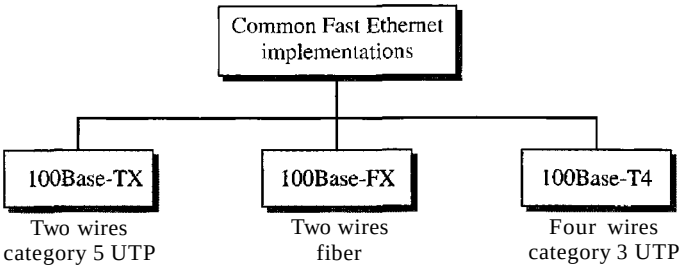
a. Point-to-point

b. Star

Implementation

Fast Ethernet implementation at the physical layer can be categorized as either two-wire or four-wire. The two-wire implementation can be either category 5 UTP (lOOBase-TX) or fiber-optic cable (lOOBase-FX). The four-wire implementation is designed only for category 3 UTP (l00Base-T4). See Figure 13.20.

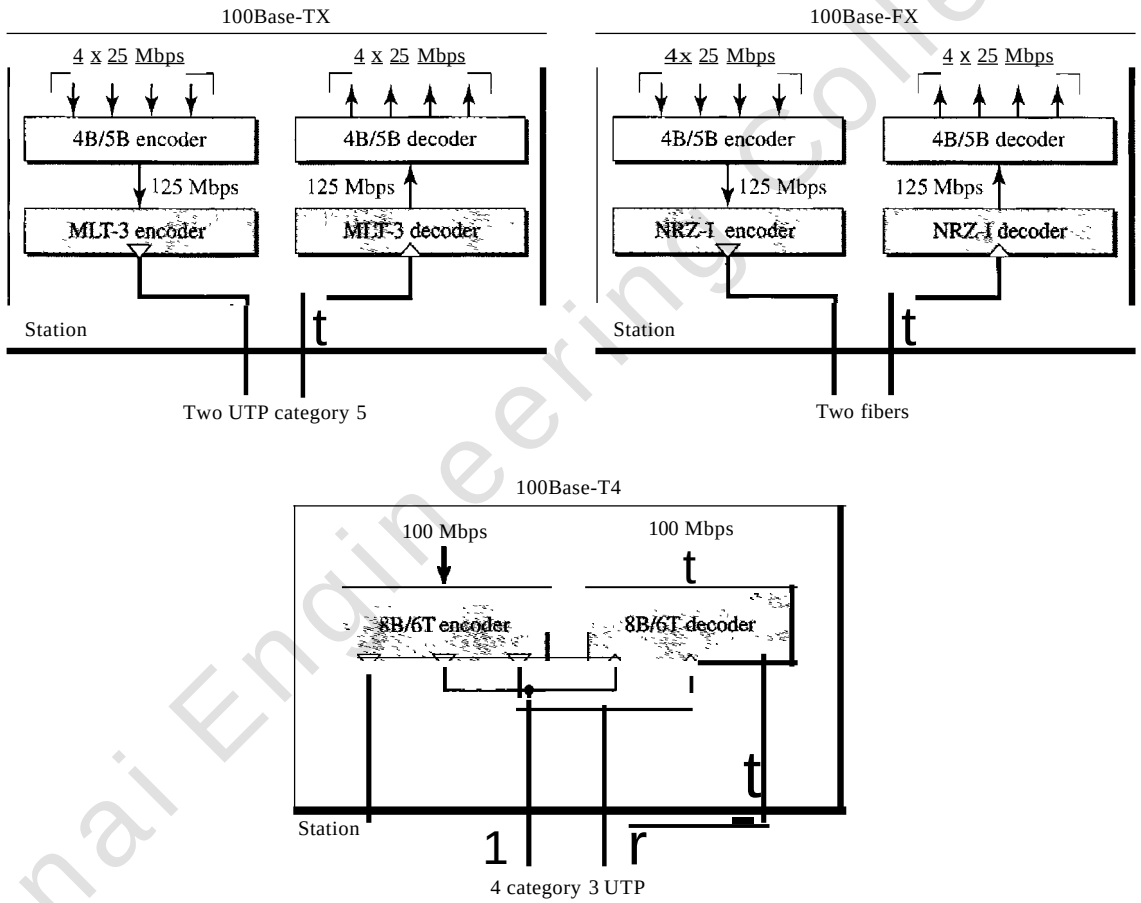
Figure 13.20 *Fast Ethernet implementations*



Encoding

Manchester encoding needs a 200-Mbaud bandwidth for a data rate of 100 Mbps, which makes it unsuitable for a medium such as twisted-pair cable. For this reason, the Fast Ethernet designers sought some alternative encoding/decoding scheme. However, it was found that one scheme would not perform equally well for all three implementations. Therefore, three different encoding schemes were chosen (see Figure 13.21).

Figure 13.21 *Encoding for Fast Ethernet implementation*



lOOBase-TX uses two pairs oftwisted-pair cable (either category 5 UTP or STP). For this implementation, the MLT-3 scheme was selected since it has good bandwidth performance (see Chapter 4). However, since MLT-3 is not a self-synchronous line coding scheme, 4B/5B block coding is used to provide bit synchronization by preventing the occurrence of a long sequence of 0s and Is (see Chapter 4). This creates a data rate of 125 Mbps, which is fed into MLT-3 for encoding.

lOOBase-FX uses two pairs of fiber-optic cables. Optical fiber can easily handle high bandwidth requirements by using simple encoding schemes. The designers of 100Base-FX selected the NRZ-I encoding scheme (see Chapter 4) for this implementation. However, NRZ-I has a bit synchronization problem for long sequences of 0s (or Is, based on the encoding), as we saw in Chapter 4. To overcome this problem, the designers used 4B/5B

block encoding as we described for IOOBase-TX. The block encoding increases the bit rate from 100 to 125 Mbps, which can easily be handled by fiber-optic cable.

A 1OOBase-TX network can provide a data rate of 100 Mbps, but it requires the use of category 5 UTP or STP cable. This is not cost-efficient for buildings that have already been wired for voice-grade twisted-pair (category 3). A new standard, called **lOOBase-T4,** was designed to use category 3 or higher UTP. The implementation uses four pairs of UTP for transmitting 100 Mbps. Encoding/decoding in 100Base-T4 is more complicated. As this implementation uses category 3 UTP, each twisted-pair cannot easily handle more than 25 Mbaud. In this design, one pair switches between sending and receiving. Three pairs of UTP category 3, however, can handle only 75 Mbaud (25 Mbaud) each. We need to use an encoding scheme that converts 100 Mbps to a 75 Mbaud signal. As we saw in Chapter 4, 8B/6T satisfies this requirement. In 8B/6T, eight data elements are encoded as six signal elements. This means that 100 Mbps uses only (6/8) x 100 Mbps, or 75 Mbaud.

Summary

Table 13.2 is a summary of the Fast Ethernet implementations.

Table 13.2 *Summary of Fast Ethernet implementations*

Characteristics	*lOOBase-TX*	*lOOBase-FX*	*100Base-T4*
Media	Cat 5 UTP or STP	Fiber	Cat 4 UTP
Number of wires	2	2	4
Maximum length	100m	100m	100m
Block encoding	4B/5B	4B/5B	
Line encoding	MLT-3	NRZ-I	8B/6T

13.5 GIGABIT ETHERNET

The need for an even higher data rate resulted in the design of the Gigabit Ethernet protocol (1000 Mbps). The IEEE committee calls the Standard 802.3z. The goals of the Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 1 Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. To support autonegotiation as defined in Fast Ethernet.

MAC **Sublayer**

A main consideration in the evolution of Ethernet was to keep the MAC sublayer untouched. However, to achieve a data rate 1 Gbps, this was no longer possible. Gigabit Ethernet has two distinctive approaches for medium access: half-duplex and full-duplex.

Almost all implementations of Gigabit Ethernet follow the full-duplex approach. However, we briefly discuss the half-duplex approach to show that Gigabit Ethernet can be compatible with the previous generations.

Full-Duplex Mode

In full-duplex mode, there is a central switch connected to all computers or other switches. In this mode, each switch has buffers for each input port in which data are stored until they are transmitted. There is no collision in this mode, as we discussed before. This means that *CSMA/CD* is not used. Lack of collision implies that the maximum length of the cable is determined by the signal attenuation in the cable, not by the collision detection process.

In the full-duplex mode of Gigabit Ethernet, there is no collision;
the maximum length of the cable is determined by the signal attenuation in the cable.

Half-Duplex Mode

Gigabit Ethernet can also be used in half-duplex mode, although it is rare. In this case, a switch can be replaced by a hub, which acts as the common cable in which a collision might occur. The half-duplex approach uses *CSMA/CD*. However, as we saw before, the maximum length of the network in this approach is totally dependent on the minimum frame size. Three methods have been defined: traditional, carrier extension, and frame bursting.

Traditional In the traditional approach, we keep the minimum length of the frame as in traditional Ethernet (512 bits). However, because the length of a bit is 11100 shorter in Gigabit Ethernet than in lO-Mbps Ethernet, the slot time for Gigabit Ethernet is 512 bits x 111000 *JlS,* which is equal to 0.512 JlS. The reduced slot time means that collision is detected 100 times earlier. This means that the maximum length of the network is 25 m. This length may be suitable if all the stations are in one room, but it may not even be long enough to connect the computers in one single office.

Carrier Extension To allow for a longer network, we increase the minimum frame length. The carrier extension approach defines the minimum length of a frame as 512 bytes (4096 bits). This means that the minimum length is 8 times longer. This method forces a station to add extension bits (padding) to any frame that is less than 4096 bits. In this way, the maximum length of the network can be increased 8 times to a length of 200 m. This allows a length of 100 m from the hub to the station.

Frame Bursting Carrier extension is very inefficient if we have a series of short frames to send; each frame carries redundant data. To improve efficiency, frame bursting was proposed. Instead of adding an extension to each frame, multiple frames are sent. However, to make these multiple frames look like one frame, padding is added between the frames (the same as that used for the carrier extension method) so that the channel is not idle. In other words, the method deceives other stations into thinking that a very large frame has been transmitted.

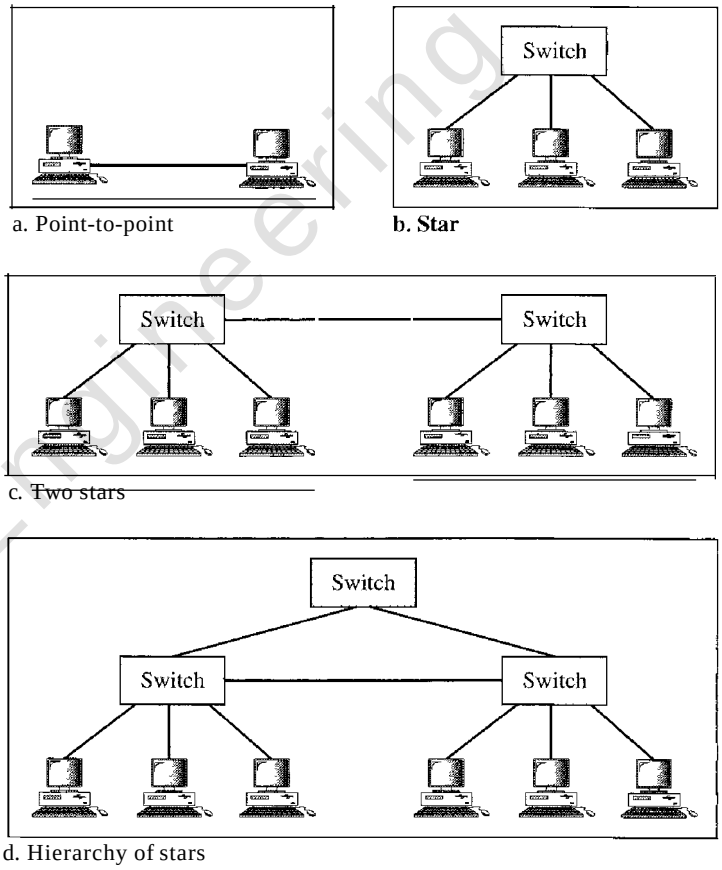
Physical Layer

The physical layer in Gigabit Ethernet is more complicated than that in Standard or Fast Ethernet. We briefly discuss some features of this layer.

Topology

Gigabit Ethernet is designed to connect two or more stations. If there are only two stations, they can be connected point-to-point. Three or more stations need to be connected in a star topology with a hub or a switch at the center. Another possible configuration is to connect several star topologies or let a star topology be part of another as shown in Figure 13.22.

Figure 13.22 *Topologies of Gigabit Ethernet*



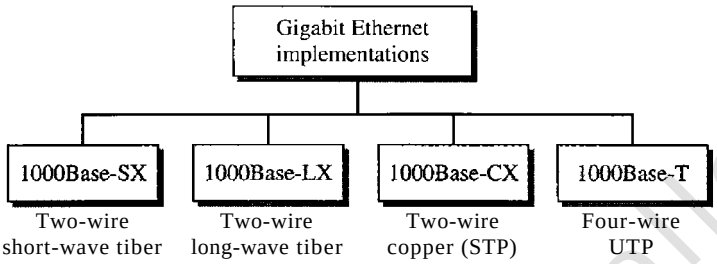
a. Point-to-point
b. Star
c. Two stars
d. Hierarchy of stars

Implementation

Gigabit Ethernet can be categorized as either a two-wire or a four-wire implementation. The two-wire implementations use fiber-optic cable (1000Base-SX, short-wave, or lOOOBase-LX, long-wave), or STP (1000Base-CX). The four-wire version uses category 5 twisted-pair cable (lOOOBase-T). In other words, we have four implementations, as shown in Figure 13.23. lOOOBase-T was designed in response to those users who

had already installed this wiring for other purposes such as Fast Ethernet or telephone serVIces.

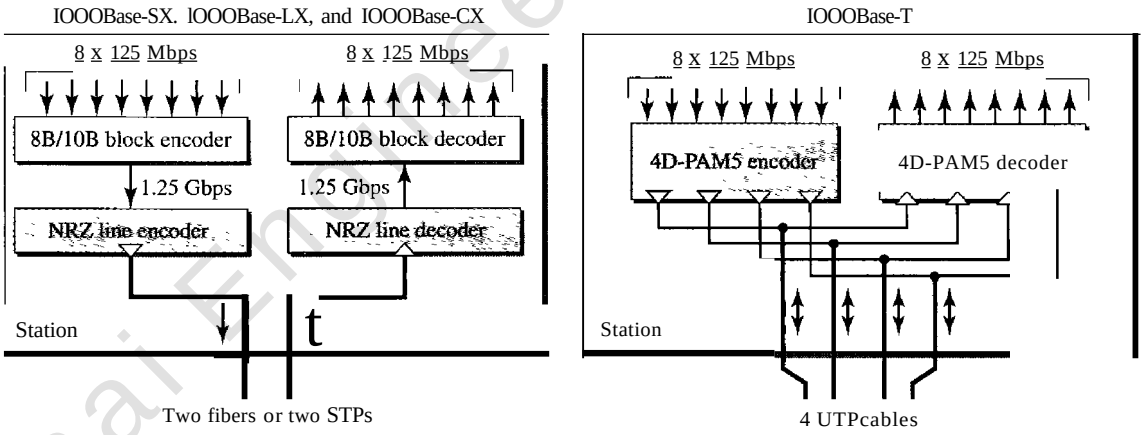
Figure 13.23 *Gigabit Ethernet implementations*



Encoding

Figure 13.24 shows the encoding/decoding schemes for the four implementations.

Figure 13.24 *Encoding in Gigabit Ethernet implementations*



Gigabit Ethernet cannot use the Manchester encoding scheme because it involves a very high bandwidth (2 GBaud). The two-wire implementations use an NRZ scheme, but NRZ does not self-synchronize properly. To synchronize bits, particularly at this high data rate, 8BIl OB block encoding, discussed in Chapter 4, is used.

This block encoding prevents long sequences of 0s or Is in the stream, but the resulting stream is 1.25 Gbps. Note that in this implementation, one wire (fiber or STP) is used for sending and one for receiving.

In the four-wire implementation it is not possible to have 2 wires for input and 2 for output, because each wire would need to carry 500 Mbps, which exceeds the capacity for category 5 UTP. As a solution, 4D-PAM5 encoding, as discussed in Chapter 4, is used to reduce the bandwidth. Thus, all four wires are involved in both input and output; each wire carries 250 Mbps, which is in the range for category 5 UTP cable_

Summary

Table 13.3 is a summary of the Gigabit Ethernet implementations.

Table 13.3 *Summary of Gigabit Ethernet implementations*

Characteristics	*lOOOBase-SX*	*lOOOBase-LX*	*lOOOBase-CX*	*lOOOBase-T*
Media	Fiber short-wave	Fiber long-wave	STP	Cat 5 UTP
Number of wires	2	2	2	4
Maximum length	550m	5000m	25m	100m
Block encoding	8B/lOB	*8B/lOB*	8B/lOB	
Line encoding	NRZ	NRZ	NRZ	4D-PAM5

Ten-Gigabit Ethernet

The IEEE committee created Ten-Gigabit Ethernet and called it Standard 802.3ae. The goals of the Ten-Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 10 Gbps.
2. Make it compatible with Standard, Fast, and Gigabit Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. Allow the interconnection of existing LANs into a metropolitan area network (MAN) or a wide area network (WAN).
7. Make Ethernet compatible with technologies such as Frame Relay and ATM (see Chapter 18).

MAC Sublayer

Ten-Gigabit Ethernet operates only in full duplex mode which means there is no need for contention; *CSMA/CD* is not used in Ten-Gigabit Ethernet.

Physical Layer

The physical layer in Ten-Gigabit Ethernet is designed for using fiber-optic cable over long distances. Three implementations are the most common: lOGBase-S, lOGBase-L, and lOGBase-E. Table 13.4 shows a summary of the Ten-Gigabit Ethernet implementaions.

Table 13.4 *Summary of Ten-Gigabit Ethernet implementations*

Characteristics	*lOGBase-S*	*lOGBase-L*	*lOGBase-E*
Media	Short-wave S50-nrn rnultimode	Long-wave 131O-nrn single mode	Extended 1550-mrn single mode
Maximum length	300m	lOkm	40km

13.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Ethernet is discussed in Chapters 10, 11, and 12 of [For03], Chapter 5 of [Kei02], Section 4.3 of [Tan03], and Chapters 15 and 16 of [Sta04]. [SpuOO] is a book about Ethernet. A complete discussion of Gigabit Ethernet can be found in [KCK98] and [Sau98]. Chapter 2 of [IzzOO] has a good comparison between different generations of Ethernet.

13.7 KEY TERMS

1000Base-CX
1000Base-LX
1000Base-SX
1000Base-T
100Base-FX
100Base-T4
100Base-TX
10Base2
lOBase5
lOBase-F
10Base-T
10GBase-E
lOGBase-L
10GBase-S
autonegotiation
bridge
carrier extension
Cheapernet
collision domain
destination service access point (DSAP)
Fast Ethernet
frame bursting
full-duplex switched Ethernet
Gigabit Ethernet
hexadecimal notation
logical link control (LLC)
media access control (MAC)
network interface card (NIC)
preamble
Project 802
source service access point (SSAP)
Standard Ethernet
switch
switched Ethernet
Ten-Gigabit Ethernet
thick Ethernet
Thicknet
thin Ethernet
transceiver
twisted-pair Ethernet

13.8 SUMMARY

- Ethernet is the most widely used local area network protocoL
- The IEEE 802.3 Standard defines I-persistent *CSMA/CD* as the access method for first-generation 10-Mbps Ethernet.
- The data link layer of Ethernet consists of the LLC sublayer and the MAC sublayer.

- The MAC sublayer is responsible for the operation of the *CSMA/CD* access method and framing.
- Each station on an Ethernet network has a unique 48-bit address imprinted on its network interface card (NIC).
- The minimum frame length for lO-Mbps Ethernet is 64 bytes; the maximum is 1518 bytes.
- The common implementations of lO-Mbps Ethernet are lOBase5 (thick Ethernet), 10Base2 (thin Ethernet), 10Base-T (twisted-pair Ethernet), and lOBase-F (fiber Ethernet).
- The 10Base5 implementation of Ethernet uses thick coaxial cable. lOBase2 uses thin coaxial cable. lOBase-T uses four twisted-pair cables that connect each station to a common hub. lOBase-F uses fiber-optic cable.
- A bridge can increase the bandwidth and separate the collision domains on an Ethernet LAN.
- A switch allows each station on an Ethernet LAN to have the entire capacity of the network to itself.
- Full-duplex mode doubles the capacity of each domain and removes the need for the *CSMA/CD* method.
- Fast Ethernet has a data rate of 100 Mbps.
- In Fast Ethernet, autonegotiation allows two devices to negotiate the mode or data rate of operation.
- The common Fast Ethernet implementations are 1OOBase-TX (two pairs of twisted-pair cable), lOOBase-FX (two fiber-optic cables), and 100Base-T4 (four pairs of voice-grade, or higher, twisted-pair cable).
- Gigabit Ethernet has a data rate of 1000 Mbps.
- Gigabit Ethernet access methods include half-duplex mode using traditional *CSMA/CD* (not common) and full-duplex mode (most popular method).
- The common Gigabit Ethernet implementations are 1000Base-SX (two optical fibers and a short-wave laser source), 1000Base-LX (two optical fibers and a long-wave laser source), and 1000Base-T (four twisted pairs).
- The latest Ethernet standard is Ten-Gigabit Ethernet that operates at 10 Gbps. The three common implementations are lOGBase-S, 10GBase-L, and 10GBase-E. These implementations use fiber-optic cables in full-duplex mode.

13.9 PRACTICE SET

Review Questions

1. How is the preamble field different from the SFD field?
2. What is the purpose of an NIC?
3. What is the difference between a unicast, multicast, and broadcast address?
4. What are the advantages of dividing an Ethernet LAN with a bridge?
5. What is the relationship between a switch and a bridge?

6. Why is there no need for *CSMA/CD* on a full-duplex Ethernet LAN?
7. Compare the data rates for Standard Ethernet, Fast Ethernet, Gigabit Ethernet, and Ten-Gigabit Ethernet.
8. What are the common Standard Ethernet implementations?
9. What are the common Fast Ethernet implementations?
10. What are the common Gigabit Ethernet implementations?
11. What are the common Ten-Gigabit Ethernet implementations?

Exercises

12. What is the hexadecimal equivalent of the following Ethernet address?

    01011010 00010001 01010101 00011000 10101010 00001111

13. How does the Ethernet address lA:2B:3CAD:5E:6F appear on the line in binary?
14. **If** an Ethernet destination address is 07:01:02:03:04:05, what is the type of the address (unicast, multicast, or broadcast)?
15. The address 43:7B:6C:DE: 10:00 has been shown as the source address in an Ethernet frame. The receiver has discarded the frame. Why?
16. An Ethernet MAC sublayer receives 42 bytes of data from the upper layer. How many bytes of padding must be added to the data?
17. An Ethernet MAC sublayer receives 1510 bytes of data from the upper layer. Can the data be encapsulated in one frame? **If** not, how many frames need to be sent? What is the size of the data in each frame?
18. What is the ratio of useful data to the entire packet for the smallest Ethernet frame? What is the ratio for the largest frame?
19. Suppose the length of a lOBase5 cable is 2500 m. **If** the speed of propagation in a thick coaxial cable is 200,000,000 m!s, how long does it take for a bit to travel from the beginning to the end of the network? Assume there are 10 μs delay in the equipment.
20. The data rate of lOBase5 is 10 Mbps. How long does it take to create the smallest frame? Show your calculation.

CHAPTER 14

Wireless LANs

Wireless communication is one of the fastest-growing technologies. The demand for connecting devices without the use of cables is increasing everywhere. Wireless LANs can be found on college campuses, in office buildings, and in many public areas.

In this chapter, we concentrate on two promising wireless technologies for LANs: IEEE 802.11 wireless LANs, sometimes called wireless Ethernet, and Bluetooth, a technology for small wireless LANs. Although both protocols need several layers to operate, we concentrate mostly on the physical and data link layers.

14.1 IEEE 802.11

IEEE has defined the specifications for a wireless LAN, called IEEE 802.11, which covers the physical and data link layers.

Architecture

The standard defines two kinds of services: the basic service set (BSS) and the extended service set (ESS).

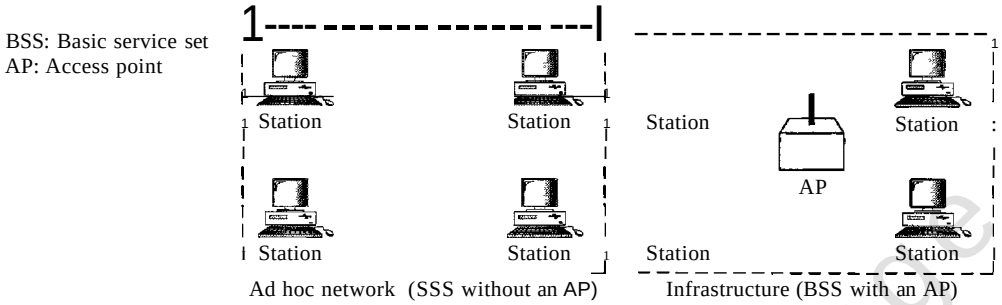
Basic Service Set

IEEE 802.11 defines the basic service set (BSS) as the building block of a wireless LAN. A basic service set is made of stationary or mobile wireless stations and an optional central base station, known as the access point (AP). Figure 14.1 shows two sets in this standard.

The BSS without an AP is a stand-alone network and cannot send data to other BSSs. It is called an *ad hoc architecture.* In this architecture, stations can form a network without the need of an AP; they can locate one another and agree to be part of a BSS. A BSS with an AP is sometimes referred to as an *infrastructure* network.

A BSS without an AP is called an ad hoc network;
a BSS with an AP is called an infrastructure network.

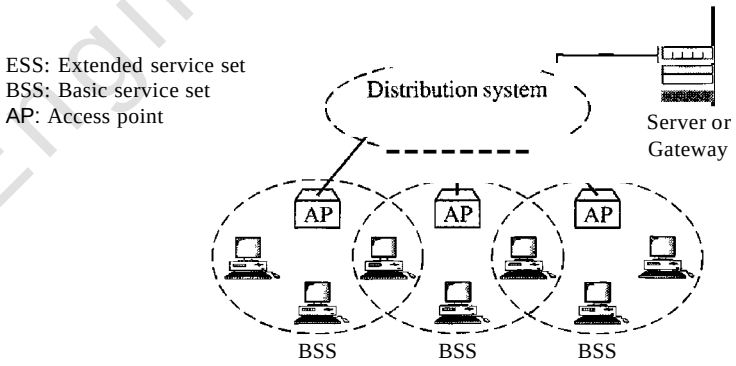
Figure 14.1 *Basic service sets (BSSs)*



Extended Service Set

An extended service set (ESS) is made up of two or more BSSs with APs. In this case, the BSSs are connected through a *distribution system,* which is usually a wired LAN. The distribution system connects the APs in the BSSs. IEEE 802.11 does not restrict the distribution system; it can be any IEEE LAN such as an Ethernet. Note that the extended service set uses two types of stations: mobile and stationary. The mobile stations are normal stations inside a BSS. The stationary stations are AP stations that are part of a wired LAN. Figure 14.2 shows an ESS.

Figure 14.2 *Extended service sets (ESSs)*



When BSSs are connected, the stations within reach of one another can communicate without the use of an AP. However, communication between two stations in two different BSSs usually occurs via two APs. The idea is similar to communication in a cellular network if we consider each BSS to be a cell and each AP to be a base station. Note that a mobile station can belong to more than one BSS at the same time.

Station Types

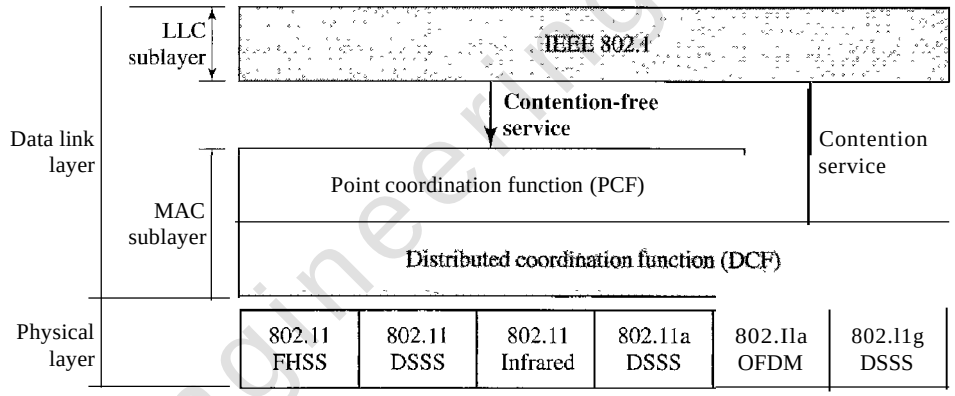
IEEE 802.11 defines three types of stations based on their mobility in a wireless LAN: no-transition, BSS·transition, and ESS-transition mobility. A station with no-transition

mobility is either stationary (not moving) or moving only inside a BSS. A station with BSS-transition mobility can move from one BSS to another, but the movement is confined inside one ESS. A station with ESS-transition mobility can move from one ESS to another. However, IEEE 802.11 does not guarantee that communication is continuous during the move.

MAC Sublayer

IEEE 802.11 defines two MAC sublayers: the distributed coordination function (DCF) and point coordination function (PCF). Figure 14.3 shows the relationship between the two MAC sublayers, the LLC sublayer, and the physical layer. We discuss the physical layer implementations later in the chapter and will now concentrate on the MAC sublayer.

Figure 14.3 *MAC layers in IEEE 802.11 standard*



Distributed Coordination Function

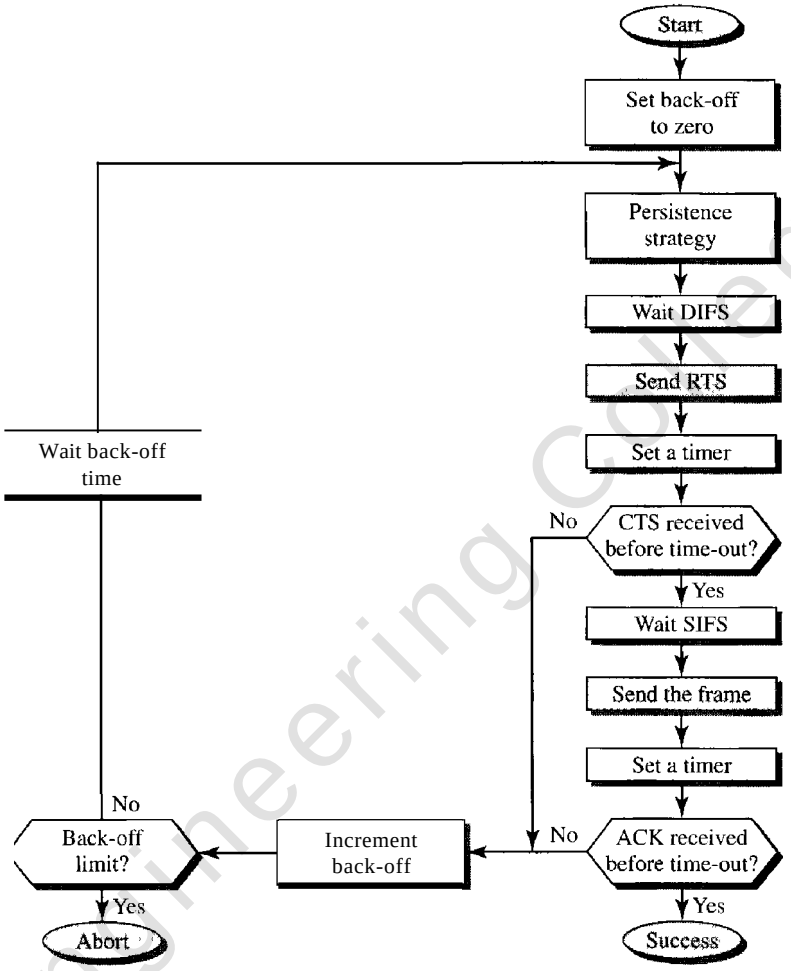
One of the two protocols defined by IEEE at the MAC sublayer is called the distributed coordination function (DCF). DCF uses *CSMA/CA* (as defined in Chapter 12) as the access method. Wireless LANs cannot implement *CSMA/CD* for three reasons:

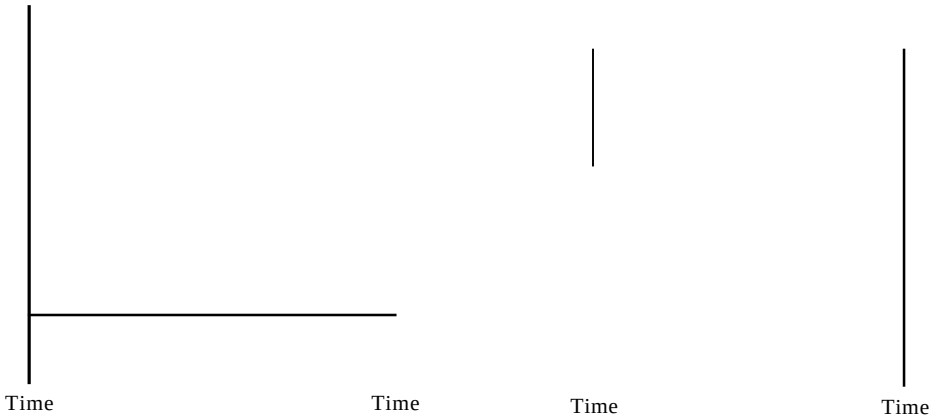
1. For collision detection a station must be able to send data and receive collision signals at the same time. This can mean costly stations and increased bandwidth requirements.
2. Collision may not be detected because of the hidden station problem. We will discuss this problem later in the chapter.
3. The distance between stations can be great. Signal fading could prevent a station at one end from hearing a collision at the other end.

Process Flowchart Figure 14.4 shows the process flowchart for *CSMA/CA* as used in wireless LANs. We will explain the steps shortly.

Frame Exchange Time Line Figure 14.5 shows the exchange of data and control frames in time.

Figure 14.4 *CSMA/CA flowchart*





1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.
   a. The channel uses a persistence strategy with back-off until the channel is idle.
   b. After the station is found to be idle, the station waits for a period of time called the distributed interframe space (DIFS); then the station sends a control frame called the request to send (RTS).
2. After receiving the RTS and waiting a period of time called the short interframe space (SIFS), the destination station sends a control frame, called the clear to send (CTS), to the source station. This control frame indicates that the destination station is ready to receive data.
3. The source station sends data after waiting an amount of time equal to SIFS.
4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in *CSMAlCD* is a kind of indication to the source that data have arrived.

Network Allocation Vector How do other stations defer sending their data if one station acquires access? In other words, how is the *collision avoidance* aspect of this protocol accomplished? The key is a feature called NAV.

When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a network allocation vector (NAV) that shows how much time must pass before these stations are allowed to check the channel for idleness. Each time a station accesses the system and sends an RTS frame, other stations start their NAV. In other words, each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired. Figure 14.5 shows the idea of NAV.

Collision During Handshaking What happens if there is collision during the time when RTS or CTS control frames are in transition, often called the handshaking period? Two or more stations may try to send RTS frames at the same time. These control frames may collide. However, because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The back-off strategy is employed, and the sender tries again.

Point Coordination Function (PCP)

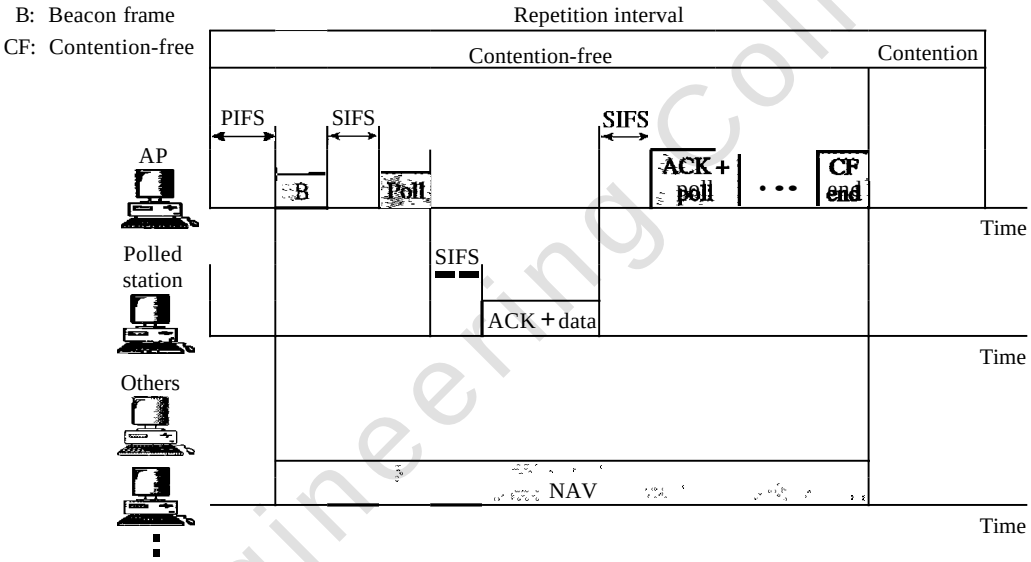
The point coordination function (PCF) is an optional access method that can be implemented in an infrastructure network (not in an ad hoc network). It is implemented on top of the DCF and is used mostly for time-sensitive transmission.

PCF has a centralized, contention-free polling access method. The AP performs polling for stations that are capable of being polled. The stations are polled one after another, sending any data they have to the AP.

To give priority to PCF over DCF, another set of interframe spaces has been defined: PIFS and SIFS. The SIFS is the same as that in DCF, but the PIFS (PCF IFS) is shorter than the DIFS. This means that if, at the same time, a station wants to use only DCF and an AP wants to use PCF, the AP has priority.

Due to the priority of PCF over DCF, stations that only use DCF may not gain access to the medium. To prevent this, a repetition interval has been designed to cover both contention-free (PCF) and contention-based (DCF) traffic. The repetition interval, which is repeated continuously, starts with a special control frame, called a beacon frame. When the stations hear the beacon frame, they start their NAV for the duration of the contention-free period of the repetition interval. Figure 14.6 shows an example of a repetition interval.

Figure 14.6 *Example of repetition interval*



During the repetition interval, the PC (point controller) can send a poll frame, receive data, send an ACK, receive an ACK, or do any combination of these (802.11 uses piggybacking). At the end of the contention-free period, the PC sends a CF end (contention-free end) frame to allow the contention-based stations to use the medium.

Fragmentation

The wireless environment is very noisy; a corrupt frame has to be retransmitted. The protocol, therefore, recommends fragmentation-the division of a large frame into smaller ones. It is more efficient to resend a small frame than a large one.

Frame Format

The MAC layer frame consists of nine fields, as shown in Figure 14.7.

- Frame control (Fe). The FC field is 2 bytes long and defines the type of frame and some control information. Table 14.1 describes the subfields. We will discuss each frame type later in this chapter.

Figure 14.7 *Frameformat*

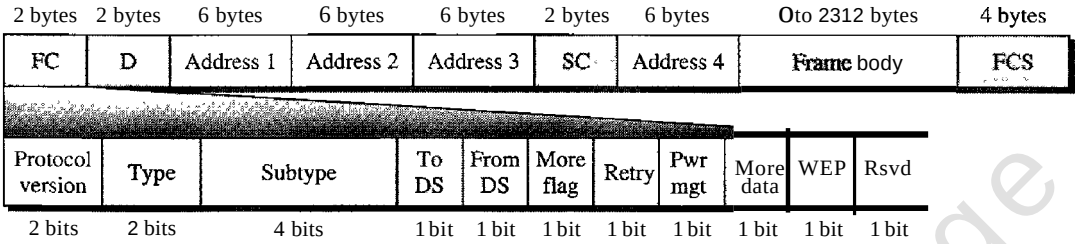


Table 14.1 *Subfields in FC field*

Field	*Explanation*
Version	Current version is 0
Type	Type of information: management (00), control (01), or data (10)
Subtype	Subtype of each type (see Table 14.2)
ToDS	Defined later
FromDS	Defined later
More flag	When set to 1, means more fragments
Retry	When set to 1, means retransmitted frame
Pwr mgt	When set to 1, means station is in power management mode
More data	When set to 1, means station has more data to send
WEP	Wired equivalent privacy (encryption implemented)
Rsvd	Reserved

- D. In all frame types except one, this field defines the duration of the transmission that is used to set the value of NAY. In one control frame, this field defines the ID of the frame.
- Addresses. There are four address fields, each 6 bytes long. The meaning of each address field depends on the value of the *To* DS and *From* DS subfields and will be discussed later.
- Sequence control. This field defines the sequence number of the frame to be used in flow control.
- Frame body. This field, which can be between 0 and 2312 bytes, contains information based on the type and the subtype defined in the FC field.
- FCS. The FCS field is 4 bytes long and contains a CRC-32 error detection sequence.

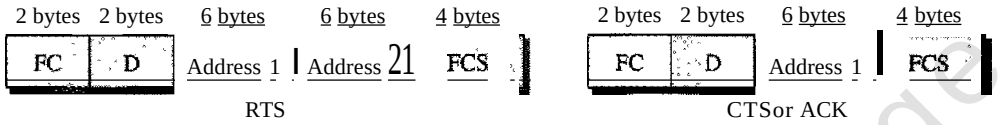
Frame Types

A wireless LAN defined by IEEE 802.11 has three categories of frames: management frames, control frames, and data frames.

Management Frames Management frames are used for the initial communication between stations and access points.

Control Frames Control frames are used for accessing the channel and acknowledging frames. Figure 14.8 shows the format.

Figure 14.8 *Controlframes*



For control frames the value of the type field is 0I; the values of the subtype fields for frames we have discussed are shown in Table 14.2.

Table 14.2 *Values ofsubfields in control frames*

Subtype	*Meaning*
1011	Request to send (RTS)
1100	Clear to send (CTS)
1101	Acknowledgment (ACK)

Data Frames Data frames are used for carrying data and control information.

Addressing Mechanism

The IEEE 802.11 addressing mechanism specifies four cases, defined by the value of the two flags in the FC field, *To DS* and *From DS.* Each flag can be either 0 or I, resulting in four different situations. The interpretation of the four addresses (address I to address 4) in the MAC frame depends on the value of these flags, as shown in Table 14.3.

Table 14.3 *Addresses*

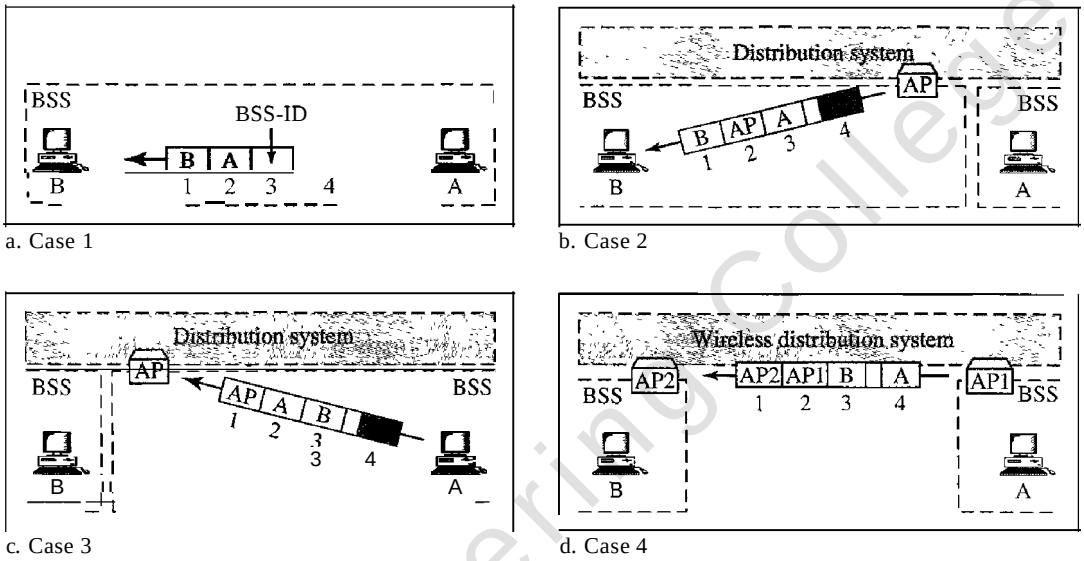
To DS	*From* DS	*Address* 1	*Address* 2	*Address* 3	*Address* 4
0	0	Destination	Source	BSS ID	N/A
0	1	Destination	SendingAP	Source	N/A
1	0	Receiving AP	Source	Destination	N/A
1	1	Receiving AP	SendingAP	Destination	Source

Note that address 1 is always the address of the next device. Address 2 is always the address of the previous device. Address 3 is the address of the final destination station if it is not defined by address I. Address 4 is the address of the original source station if it is not the same as address 2.

- Case 1: 00 In this case, *To DS* = 0 and *From DS* = 0. This means that the frame is not going to a distribution system *(To DS* = 0) and is not coming from a distribution

system *(From DS* = 0). The frame is going from one station in a BSS to another without passing through the distribution system. The ACK frame should be sent to the original sender. The addresses are shown in Figure 14.9.

Figure 14.9 *Addressing mechanisms*



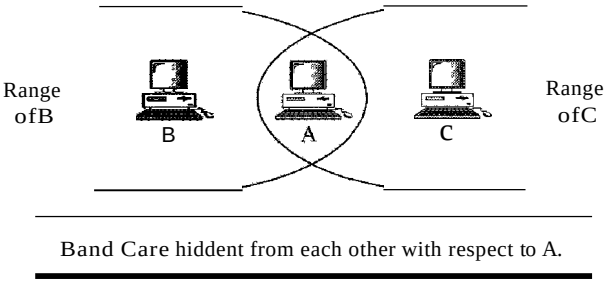
- Case 2: 01 In this case, *To DS* = 0 and *From DS* = 1. This means that the frame is coming from a distribution system *(From* DS = 1). The frame is coming from an AP and going to a station. The ACK should be sent to the AP. The addresses are as shown in Figure 14.9. Note that address 3 contains the original sender of the frame (in another BSS).
- Case 3: 10 In this case, *To DS* = 1 and *From DS* = 0. This means that the frame is going to a distribution system *(To DS* = 1). The frame is going from a station to an AP. The ACK is sent to the original station. The addresses are as shown in Figure 14.9. Note that address 3 contains the final destination of the frame (in another BSS).
- Case 4:11 In this case, *To DS* = 1 and *From DS* = 1. This is the case in which the distribution system is also wireless. The frame is going from one AP to another AP in a wireless distribution system. We do not need to define addresses if the distribution system is a wired LAN because the frame in these cases has the format of a wired LAN frame (Ethernet, for example). Here, we need four addresses to define the original sender, the final destination, and two intermediate APs. Figure 14.9 shows the situation.

Hidden and Exposed Station Problems

We referred to hidden and exposed station problems in the previous section. It is time now to dicuss these problems and their effects.

Hidden Station Problem Figure 14.10 shows an example of the hidden station problem. Station B has a transmission range shown by the left oval (sphere in space); every station in this range can hear any signal transmitted by station B. Station C has

Figure 14.10 *Hidden station problem*



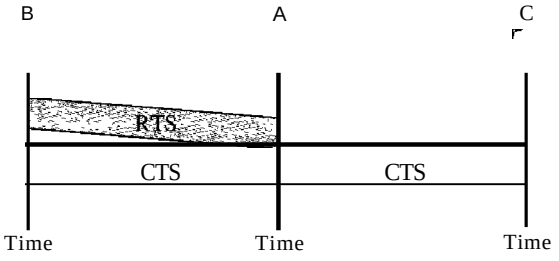
a transmission range shown by the right oval (sphere in space); every station located in this range can hear any signal transmitted by C. Station C is outside the transmission range of B; likewise, station B is outside the transmission range of C. Station A, however, is in the area covered by both Band C; it can hear any signal transmitted by B or C.

Assume that station B is sending data to station A. In the middle of this transmission, station C also has data to send to station A. However, station C is out of B's range and transmissions from B cannot reach C. Therefore C thinks the medium is free. Station C sends its data to A, which results in a collision at A because this station is receiving data from both B and C. In this case, we say that stations Band C are hidden from each other with respect to A. Hidden stations can reduce the capacity of the network because of the possibility of collision.

The solution to the hidden station problem is the use of the handshake frames (RTS and CTS) that we discussed earlier. Figure 14.11 shows that the RTS message from B reaches A, but not C. However, because both Band C are within the range of A, the CTS message, which contains the duration of data transmission from B to A reaches C. Station C knows that some hidden station is using the channel and refrains from transmitting until that duration is over.

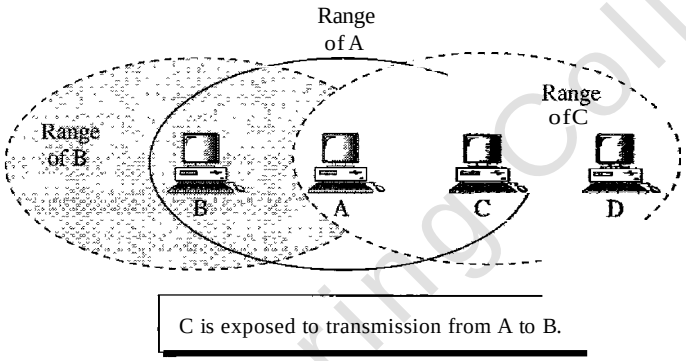
The CTS frame in CSMAICA handshake can prevent collision from a hidden station.

Figure 14.11 *Use of handshaking to prevent hidden station problem*



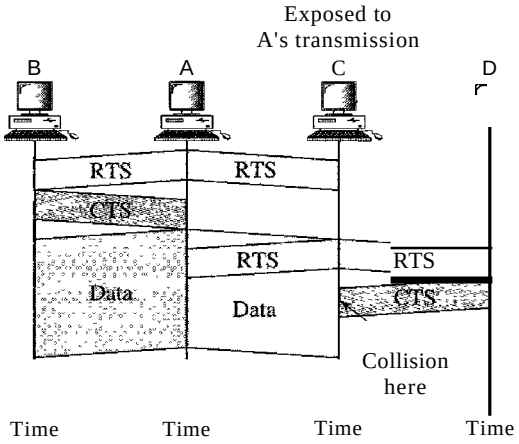
Exposed Station Problem Now consider a situation that is the inverse of the previous one: the exposed station problem. In this problem a station refrains from using a channel when it is, in fact, available. In Figure 14.12, station A is transmitting to station B. Station C has some data to send to station D, which can be sent without interfering with the transmission from A to B. However, station C is exposed to transmission from A; it hears what A is sending and thus refrains from sending. In other words, C is too conservative and wastes the capacity of the channel.

Figure 14.12 *Exposed station problem*



The handshaking messages RTS and CTS cannot help in this case, despite what you might think. Station C hears the RTS from A, but does not hear the CTS from B. Station C, after hearing the RTS from A, can wait for a time so that the CTS from B reaches A; it then sends an RTS to D to show that it needs to communicate with D. Both stations B and A may hear this RTS, but station A is in the sending state, not the receiving state. Station B, however, responds with a CTS. The problem is here. If station A has started sending its data, station C cannot hear the CTS from station D because of the collision; it cannot send its data to D. It remains exposed until A finishes sending its data as Figure 14.13 shows.

Figure 14.13 *Use of handshaking in exposed station problem*



Physical Layer

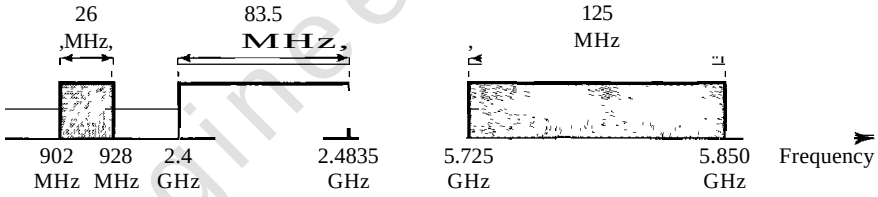
We discuss six specifications, as shown in Table 14.4.

Table 14.4 *Physical layers*

IEEE	*Technique*	*Band*	*Modulation*	*Rate (Mbps)*
802.11	FHSS	2.4 GHz	FSK	1 and 2
	DSSS	2.4 GHz	PSK	1 and 2
		Infrared	PPM	1 and 2
802.11a	OFDM	5.725 GHz	PSKorQAM	6 to 54
802.l1b	DSSS	2.4 GHz	PSK	5.5 and 11
802.1Ig	OFDM	2.4 GHz	Different	22 and 54

All implementations, except the infrared, operate in the *industrial, scientific,* and *medical (ISM)* band, which defines three unlicensed bands in the three ranges 902-928 MHz, 2.400--4.835 GHz, and 5.725-5.850 GHz, as shown in Figure 14.14.

Figure 14.14 *industrial, scientific, and medical (ISM) band*



IEEE 802.11 FHSS

IEEE 802.11 FHSS uses the frequency-hopping spread spectrum (FHSS) method as discussed in Chapter 6. FHSS uses the 2.4-GHz ISM band. The band is divided into 79 subbands of 1 MHz (and some guard bands). A pseudorandom number generator selects the hopping sequence. The modulation technique in this specification is either two-level FSK or four-level FSK with I or 2 bitslbaud, which results in a data rate of 1 or 2 Mbps, as shown in Figure 14.15.

IEEE 802.11 DSSS

IEEE 802.11 DSSS uses the direct sequence spread spectrum (DSSS) method as discussed in Chapter 6. DSSS uses the 2.4-GHz ISM band. The modulation technique in this specification is PSK at 1 Mbaud/s. The system allows 1 or 2 bitslbaud (BPSK or QPSK), which results in a data rate of 1 or 2 Mbps, as shown in Figure 14.16.

IEEE 802.11 Infrared

IEEE 802.11 infrared uses infrared light in the range of 800 to 950 nm. The modulation technique is called pulse position modulation (PPM). For a I-Mbps data rate, a 4-bit

Figure 14.15 *Physical layer of IEEE 802.11 FHSS*

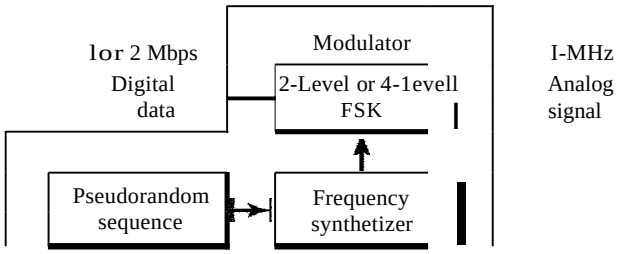
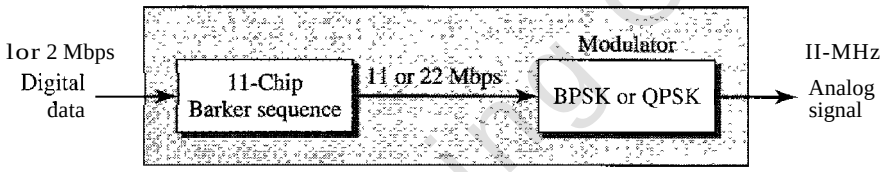
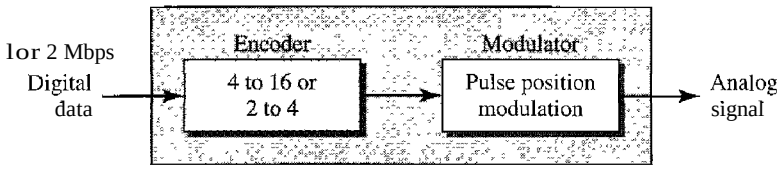


Figure 14.16 *Physical layer of IEEE 802.11 DSSS*



sequence is first mapped into a 16-bit sequence in which only one bit is set to 1 and the rest are set to 0. For a 2-Mbps data rate, a 2-bit sequence is first mapped into a 4-bit sequence in which only one bit is set to 1 and the rest are set to 0. The mapped sequences are then converted to optical signals; the presence of light specifies 1, the absence of light specifies 0. See Figure 14.17.

Figure 14.17 *Physical layer of IEEE 802.11 infrared*



IEEE 802.lla OFDM

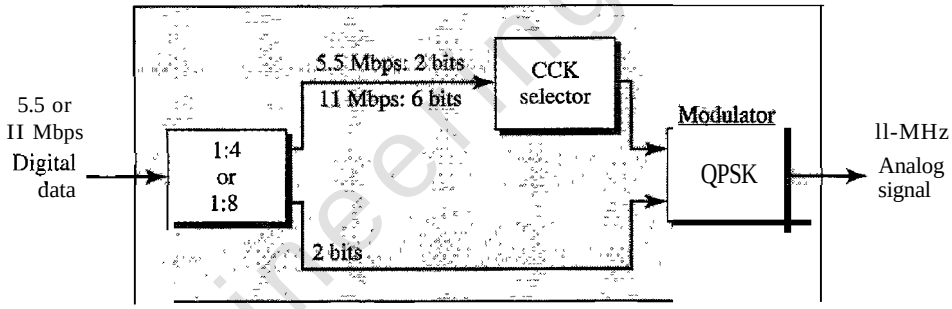
IEEE 802.Ila OFDM describes the orthogonal frequency-division multiplexing (OFDM) method for signal generation in a 5-GHz ISM band. OFDM is similar to FDM as discussed in Chapter 6, with one major difference: All the subbands are used by one source at a given time. Sources contend with one another at the data link layer for access. The band is divided into 52 subbands, with 48 subbands for sending 48 groups of bits at a time and 4 subbands for control information. The scheme is similar to ADSL, as discussed in Chapter 9. Dividing the band into subbands diminishes the effects of interference. If the subbands are used randomly, security can also be increased.

OFDM uses PSK and QAM for modulation. The common data rates are 18 Mbps (PSK) and 54 Mbps (QAM).

IEEE 802.llb DSSS

IEEE 802.11b DSSS describes the high-rate direct sequence spread spectrum (HR-DSSS) method for signal generation in the 2.4-GHz ISM band. HR-DSSS is similar to DSSS except for the encoding method, which is called complementary code keying (CCK). CCK encodes 4 or 8 bits to one CCK symbol. To be backward compatible with DSSS, HR-DSSS defines four data rates: 1,2, 5.5, and 11 Mbps. The first two use the same modulation techniques as DSSS. The 5.5-Mbps version uses BPSK and transmits at 1.375 Mbaudls with 4-bit CCK encoding. The II-Mbps version uses QPSK and transmits at 1.375 Mbps with 8-bit CCK encoding. Figure 14.18 shows the modulation technique for this standard.

Figure 14.18 *Physical layer of IEEE 802.11b*



IEEE 802.11g

This new specification defines forward error correction and OFDM using the 2.4-GHz ISM band. The modulation technique achieves a 22- or 54-Mbps data rate. It is backward-compatible with 802.11b, but the modulation technique is OFDM.

14.2 BLUETOOTH

Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers (desktop and laptop), cameras, printers, coffee makers, and so on. A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously; the devices, sometimes called gadgets, find each other and make a network called a piconet. A Bluetooth LAN can even be connected to the Internet if one of the gadgets has this capability. A Bluetooth LAN, by nature, cannot be large. If there are many gadgets that try to connect, there is chaos.

Bluetooth technology has several applications. Peripheral devices such as a wireless mouse or keyboard can communicate with the computer through this technology. Monitoring devices can communicate with sensor devices in a small health care center. Home security devices can use this technology to connect different sensors to the main

security controller. Conference attendees can synchronize their laptop computers at a conference.

Bluetooth was originally started as a project by the Ericsson Company. It is named for Harald Blaatand, the king of Denmark (940-981) who united Denmark and Norway. *Blaatand* translates to *Bluetooth* in English.

Today, Bluetooth technology is the implementation of a protocol defined by the IEEE 802.15 standard. The standard defines a wireless personal-area network (PAN) operable in an area the size of a room or a hall.

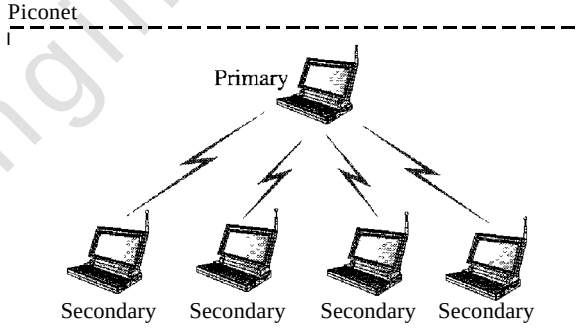
Architecture

Bluetooth defines two types of networks: piconet and scatternet.

Piconets

A Bluetooth network is called a piconet, or a small net. A piconet can have up to eight stations, one of which is called the primary;t the rest are called secondaries. All the secondary stations synchronize their clocks and hopping sequence with the primary. Note that a piconet can have only one primary station. The communication between the primary and the secondary can be one-to-one or one-to-many. Figure 14.19 shows a piconet.

Figure 14.19 *Piconet*



Although a piconet can have a maximum of seven secondaries, an additional eight secondaries can be in the *parked state.* A secondary in a parked state is synchronized with the primary, but cannot take part in communication until it is moved from the parked state. Because only eight stations can be active in a piconet, activating a station from the parked state means that an active station must go to the parked state.

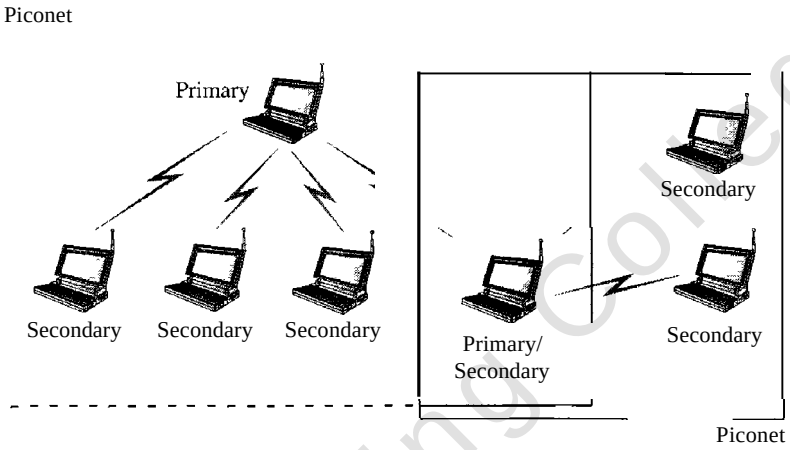
Scat/ernet

Piconets can be combined to form what is called a scatternet. A secondary station in one piconet can be the primary in another piconet. This station can receive messages

tThe literature sometimes uses the terms *master* and *slave* instead of *primary* and *secondary.* We prefer the latter.

from the primary in the first piconet (as a secondary) and, acting as a primary, deliver them to secondaries in the second piconet. A station can be a member of two piconets. Figure 14.20 illustrates a scatternet.

Figure 14.20 *Scatternet*



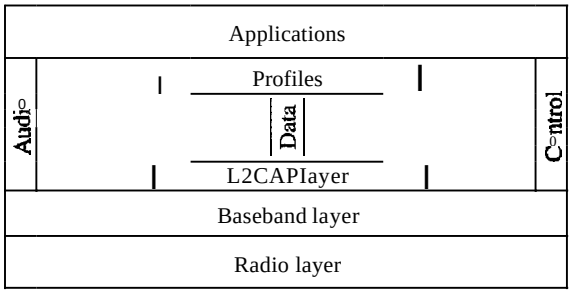
Bluetooth Devices

A Bluetooth device has a built-in short-range radio transmitter. The current data rate is 1 Mbps with a 2.4-GHz bandwidth. This means that there is a possibility of interference between the IEEE 802.11b wireless LANs and Bluetooth LANs.

Bluetooth Layers

Bluetooth uses several layers that do not exactly match those of the Internet model we have defined in this book. Figure 14.21 shows these layers.

Figure 14.21 *Bluetooth layers*



Radio Layer

The radio layer is roughly equivalent to the physical layer of the Internet model. Bluetooth devices are low-power and have a range of 10 m.

Band

Bluetooth uses a 2.4-GHz ISM band divided into 79 channels of 1 MHz each.

FHSS

Bluetooth uses the frequency-hopping spread spectrum (FHSS) method in the physical layer to avoid interference from other devices or other networks. Bluetooth hops 1600 times per second, which means that each device changes its modulation frequency 1600 times per second. A device uses a frequency for only 625 μs (1/1600 s) before it hops to another frequency; the dwell time is 625 μs.

Modulation

To transform bits to a signal, Bluetooth uses a sophisticated version of FSK, called GFSK (FSK with Gaussian bandwidth filtering; a discussion of this topic is beyond the scope of this book). GFSK has a can'ier frequency. Bit 1 is represented by a frequency deviation above the carrier; bit a is represented by a frequency deviation below the carrier. The frequencies, in megahertz, are defined according to the following formula for each channel:

$$fc = 2402 + n \qquad n = 0, 1, 2, 3, \ldots, 78$$

For example, the first channel uses carrier frequency 2402 MHz (2.402 GHz), and the second channel uses carrier frequency 2403 MHz (2.403 GHz).

Baseband Layer

The baseband layer is roughly equivalent to the MAC sublayer in LANs. The access method is TDMA (see Chapter 12). The primary and secondary communicate with each other using time slots. The length of a time slot is exactly the same as the dwell time, 625 μs. This means that during the time that one frequency is used, a sender sends a frame to a secondary, or a secondary sends a frame to the primary. Note that the communication is only between the primary and a secondary; secondaries cannot communicate directly with one another.

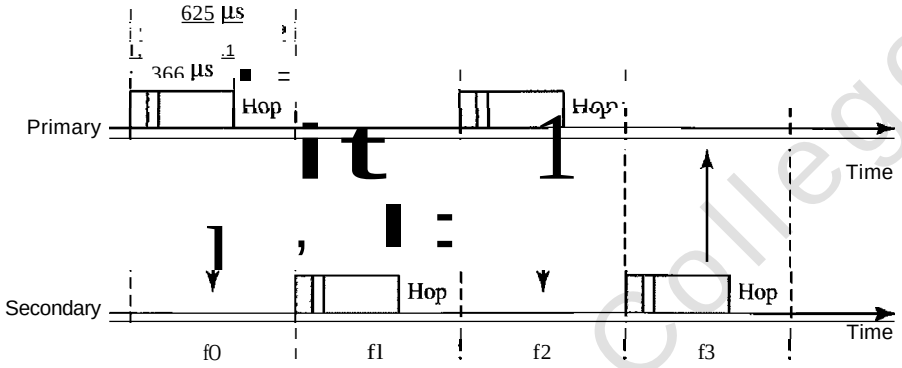
TDMA

Bluetooth uses a form of TDMA (see Chapter 12) that is called TDD-TDMA (time-division duplex TDMA). TDD-TDMA is a kind of half-duplex communication in which the secondary and receiver send and receive data, but not at the same time (half-duplex); however, the communication for each direction uses different hops. This is similar to walkie-talkies using different carrier frequencies.

Single-Secondary Communication If the piconet has only one secondary, the TDMA operation is very simple. The time is divided into slots of 625 μs. The primary uses even-numbered slots (0, 2, 4, ...); the secondary uses odd-numbered slots (1, 3, 5, ...). TDD-TDMA allows the primary and the secondary to communicate in half-duplex mode.

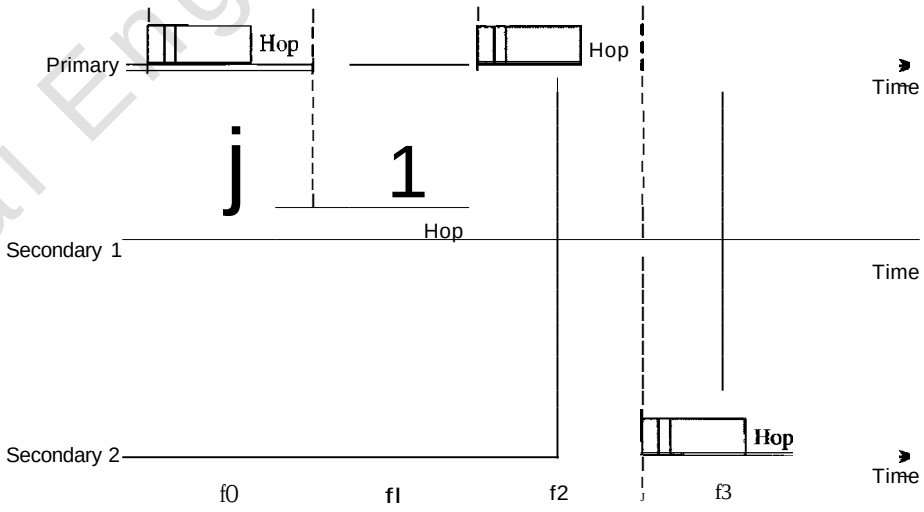
In slot 0, the primary sends, and the secondary receives; in slot 1, the secondary sends, and the primary receives. The cycle is repeated. Figure 14.22 shows the concept.

Figure 14.22 *Single-secondary communication*



Multiple-Secondary Communication The process is a little more involved if there is more than one secondary in the piconet. Again, the primary uses the even-numbered slots, but a secondary sends in the next odd-numbered slot if the packet in the previous slot was addressed to it. All secondaries listen on even-numbered slots, but only one secondary sends in any odd-numbered slot. Figure 14.23 shows a scenario.

Figure 14.23 *Multiple-secondary communication*



Let us elaborate on the figure.

1. In slot 0, the primary sends a frame to secondary 1.
2. In slot 1, only secondary I sends a frame to the primary because the previous frame was addressed to secondary 1; other secondaries are silent.

3. In slot 2, the primary sends a frame to secondary 2.
4. In slot 3, only secondary 2 sends a frame to the primary because the previous frame was addressed to secondary 2; other secondaries are silent.
5. The cycle continues.

We can say that this access method is similar to a poll/select operation with reservations. When the primary selects a secondary, it also polls it. The next time slot is reserved for the polled station to send its frame. If the polled secondary has no frame to send, the channel is silent.

Physical Links

Two types of links can be created between a primary and a secondary: SCQ links and ACL links.

sca A synchronous connection-oriented (SeQ) link is used when avoiding latency (delay in data delivery) is more important than integrity (error-free delivery). In an SCQ link, a physical link is created between the primary and a secondary by reserving specific slots at regular intervals. The basic unit of connection is two slots, one for each direction. If a packet is damaged, it is never retransmitted. SCQ is used for real-time audio where avoiding delay is all-important. A secondary can create up to three SCQ links with the primary, sending digitized audio (PCM) at 64 kbps in each link.

ACL An asynchronous connectionless link (ACL) is used when data integrity is more important than avoiding latency. In this type of link, if a payload encapsulated in the frame is corrupted, it is retransmitted. A secondary returns an ACL frame in the available odd-numbered slot if and only if the previous slot has been addressed to it. ACL can use one, three, or more slots and can achieve a maximum data rate of 721 kbps.

Frame Format

A frame in the baseband layer can be one of three types: one-slot, three-slot, or five-slot. A slot, as we said before, is 625 μs. However, in a one-slot frame exchange, 259 μs is needed for hopping and control mechanisms. This means that a one-slot frame can last only 625 - 259, or 366 μs. With a I-MHz bandwidth and 1 bit/Hz, the size of a one-slot frame is 366 bits.

A three-slot frame occupies three slots. However, since 259 μs is used for hopping, the length of the frame is 3 x 625 - 259 = 1616 μs or 1616 bits. A device that uses a three-slot frame remains at the same hop (at the same carrier frequency) for three slots. Evcn though only onc hop numbcr is used, three hop numbers are consumed. That means the hop number for each frame is equal to the first slot of the frame.

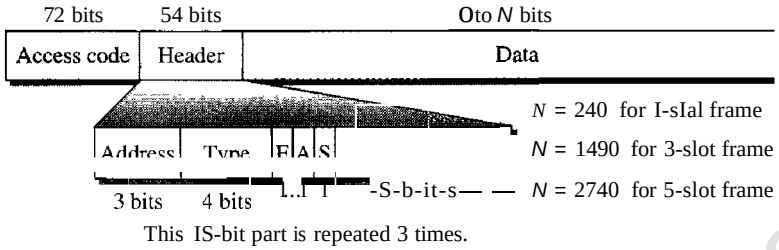
A five-slot frame also uses 259 bits for hopping, which means that the length of the frame is 5 x 625 - 259 = 2866 bits.

Figure 14.24 shows the format of the three frame types.

The following describes each field:

- Access code. This 72-bit field normally contains synchronization bits and the identifier of the primary to distinguish the frame of one piconet from another.

Figure 14.24 *Framefannat types*



- Header. This 54-bit field is a repeated I8-bit pattern. Each pattern has the following subfields:
  1. Address. The 3-bit address subfield can define up to seven secondaries (I to 7). If the address is zero, it is used for broadcast communication from the primary to all secondaries.
  2. Type. The 4-bit type subfield defines the type of data coming from the upper layers. We discuss these types later.
  3. F. This I-bit subfield is for flow control. When set (I), it indicates that the device is unable to receive more frames (buffer is full).
  4. A. This I-bit subfield is for acknowledgment. Bluetooth uses Stop-and-Wait ARQ; I bit is sufficient for acknowledgment.
  5. S. This I-bit subfield holds a sequence number. Bluetooth uses Stop-and-Wait ARQ; I bit is sufficient for sequence numbering.
  6. HEC. The 8-bit header error correction subfield is a checksum to detect errors in each 18-bit header section.

  The header has three identical 18-bit sections. The receiver compares these three sections, bit by bit. If each of the corresponding bits is the same, the bit is accepted; if not, the majority opinion rules. This is a form of forward error correction (for the header only). This double error control is needed because the nature of the communication, via air, is very noisy. Note that there is no retransmission in this sublayer.
- Payload. This subfield can be 0 to 2740 bits long. It contains data or control information corning from the upper layers.

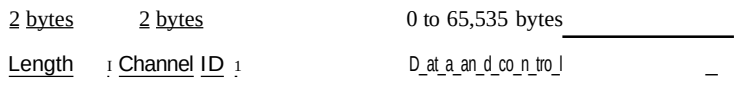
L2CAP

The Logical Link Control and Adaptation Protocol, or L2CAP (L2 here means LL), is roughly equivalent to the LLC sublayer in LANs. It is used for data exchange on an ACL link; SCQ channels do not use L2CAP. Figure 14.25 shows the format of the data packet at this level.

The I6-bit length field defines the size of the data, in bytes, coming from the upper layers. Data can be up to 65,535 bytes. The channel ID (CID) defines a unique identifier for the virtual channel created at this level (see below).

The L2CAP has specific duties: multiplexing, segmentation and reassembly, quality of service (QoS), and group management.

Figure 14.25 *L2CAP data packetfarmat*



Multiplexing

The L2CAP can do multiplexing. At the sender site, it accepts data from one of the upper-layer protocols, frames them, and delivers them to the baseband layer. At the receiver site, it accepts a frame from the baseband layer, extracts the data, and delivers them to the appropriate protocol layer. It creates a kind of virtual channel that we will discuss in later chapters on higher-level protocols.

Segmentation and Reassembly

The maximum size of the payload field in the baseband layer is 2774 bits, or 343 bytes. This includes 4 bytes to define the packet and packet length. Therefore, the size of the packet that can arrive from an upper layer can only be 339 bytes. However, application layers sometimes need to send a data packet that can be up to 65,535 bytes (an Internet packet, for example). The L2CAP divides these large packets into segments and adds extra information to define the location of the segments in the original packet. The L2CAP segments the packet at the source and reassembles them at the destination.

QoS

Bluetooth allows the stations to define a quality-of-service level. We discuss quality of service in Chapter 24. For the moment, it is sufficient to know that if no quality-of-service level is defined, Bluetooth defaults to what is called *best-effort* service; it will do its best under the circumstances.

Group Management

Another functionality of L2CAP is to allow devices to create a type of logical addressing between themselves. This is similar to multicasting. For example, two or three secondary devices can be part of a multicast group to receive data from the primary.

Other Upper Layers

Bluetooth defines several protocols for the upper layers that use the services of L2CAP; these protocols are specific for each purpose.

14.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

Wireless LANs and Bluetooth are discussed in several books including [Sch03] and [Gas02]. Wireless LANs are discussed in Chapter 15 of [For03], Chapter 17 of [Sta04], Chapters 13 and 14 of [Sta02], and Chapter 8 of [Kei021. Bluetooth is discussed in Chapter 15 of [Sta02] and Chapter 15 of [For03].

14.4 KEY TERMS

access point (AP)
asynchronous connectionless link (ACL)
basic service set (BSS)
beacon frame
Bluetooth
BSS-transition mobility
complementary code keying (CCK)
direct sequence spread spectrum (OSSS)
distributed coordination function (OCF)
distributed interframe space (OIFS)
ESS-transition mobility
extended service set (ESS)
frequency-hopping spread spectrum (FHSS)
handshaking period
high-rate direct sequence spread spectrum (HR-OSSS)
IEEE 802.11
Logical Link Control and Adaptation Protocol (L2CAP)
network allocation vector (NAV)
no-transition mobility
orthogonal frequency-division multiplexing (OFOM)
piconet
point coordination function (PCF)
primary
pulse position modulation (PPM)
repetition interval
scatternet
secondary
short interframe space (SIFS)
synchronous connection-oriented (SCO)
TDD-TOMA (time-division duplex TOMA)
wireless LAN

14.5 SUMMARY

- The IEEE 802.11 standard for wireless LANs defines two services: basic service set (BSS) and extended service set (ESS).
- The access method used in the distributed coordination function (OCF) MAC sub-layer is *CSMAICA.*
- The access method used in the point coordination function (PCF) MAC sublayer is polling.
- The network allocation vector (NAV) is a timer used for collision avoidance.
- The MAC layer frame has nine fields. The addressing mechanism can include up to four addresses.
- Wireless LANs use management frames, control frames, and data frames.

- IEEE 802.11 defines several physical layers, with different data rates and modulating techniques.
- Bluetooth is a wireless LAN technology that connects devices (called gadgets) in a small area.
- A Bluetooth network is called a piconet. Multiple piconets fonn a network called a scatternet.
- A Bluetooth network consists of one primary device and up to seven secondary devices.

14.6 PRACTICE SET

Review Questions

1. What is the difference between a BSS and an ESS?
2. Discuss the three types of mobility in a wireless LAN.
3. How is OFDM different from FDM?
4. What is the access method used by wireless LANs?
5. What is the purpose of the NAV?
6. Compare a piconet and a scatternet.
7. Match the layers in Bluetooth and the Internet model.
8. What are the two types of links between a Bluetooth primary and a Bluetooth secondary?
9. In multiple-secondary communication, who uses the even-numbered slots and who uses the odd-numbered slots?
10. How much time in a Bluetooth one-slot frame is used for the hopping mechanism? What about a three-slot frame and a five-slot frame?

Exercises

1]. Compare and contrast *CSMAlCD* with *CSMAICA.*

12. Use Table 14.5 to compare and contrast the fields in IEEE 802.3 and 802.11.

Table **14.5** *Exercise 12*

Fields	*IEEE 802.3 Field Size*	*IEEE 802.11 Field Size*
Destination address		
Source address		
Address 1		
Address 2		
Address 3		
Address 4		
FC		

Table 14.5 *Exercise 12 (continued)*

Fields	*IEEE 802.3 Field Size*	*IEEE 802.11 Field Size*
D/ID		
SC		
PDU length		
Data and padding		
Frame body		
FCS (CRC)		

CHAPTER 15

Connecting LANs, Backbone Networks, and Virtual LANs

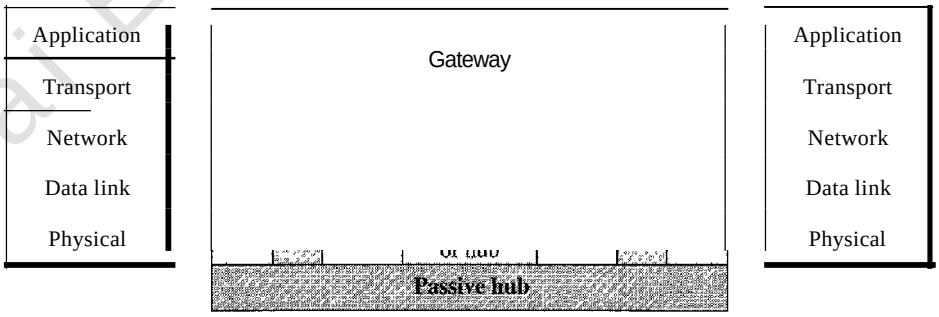
LANs do not normally operate **in** isolation. They are connected to one another or to the Internet. To connect LANs, or segments of LANs, we use connecting devices. Connecting devices can operate **in** different layers of the Internet model. **In** this chapter, we discuss only those that operate **in** the physical and data link layers; we discuss those that operate in the first three layers in Chapter 19.

After discussing some connecting devices, we show how they are used to create backbone networks. Finally, we discuss virtual local area networks (VLANs).

15.1 CONNECTING DEVICES

In this section, we divide **connecting devices** into five different categories based on the layer **in** which they operate **in** a network, as shown **in** Figure 15.1.

Figure 15.1 *Five categories of connecting devices*



The five categories contain devices which can be defined as

1. Those which operate below the physical layer such as a passive hub.
2. Those which operate at the physical layer (a repeater or an active hub).
3. Those which operate at the physical and data link layers (a bridge or a two-layer switch).

4. Those which operate at the physical, data link, and network layers (a router or a three-layer switch).
5. Those which can operate at all five layers (a gateway).

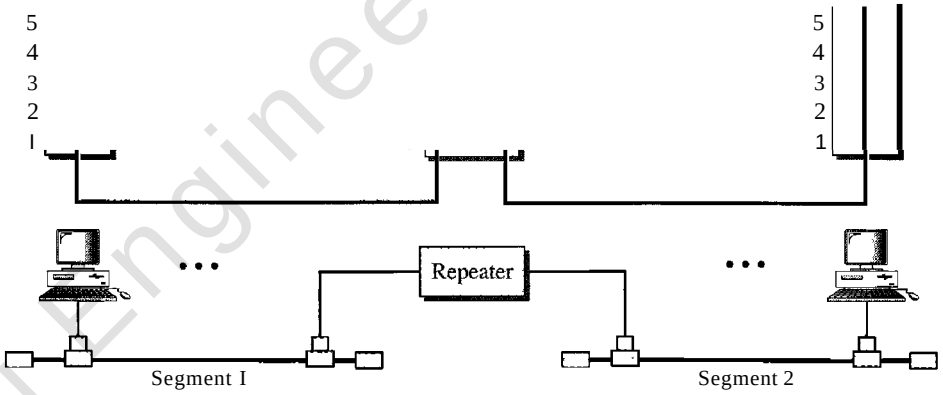
Passive Hubs

A passive hub is just a connector. It connects the wires coming from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point. This type of a hub is part of the media; its location in the Internet model is below the physical layer.

Repeaters

A repeater is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern. The repeater then sends the refreshed signal. A repeater can extend the physical length of a LAN, as shown in Figure 15.2.

Figure 15.2 *A repeater connecting two segments of a LAN*



A repeater does not actually connect two LANs; it connects two segments of the same LAN. The segments connected are still part of one single LAN. A repeater is not a device that can connect two LANs of different protocols.

A repeater connects segments of a LAN.

A repeater can overcome the 10Base5 Ethernet length restriction. In this standard, the length of the cable is limited to 500 m. To extend this length, we divide the cable into segments and install repeaters between segments. Note that the whole network is still considered one LAN, but the portions of the network separated by repeaters are called segments. The repeater acts as a two-port node, but operates only in the physical layer. When it receives a frame from any of the ports, it regenerates and forwards it to the other port.

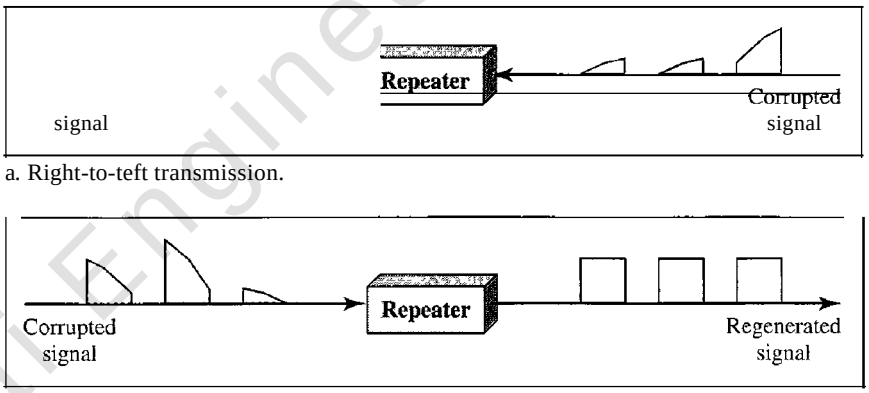
A repeater forwards every frame; it has no filtering capability.

It is tempting to compare a repeater to an amplifier, but the comparison is inaccurate. An amplifier cannot discriminate between the intended signal and noise; it amplifies equally everything fed into it. A repeater does not amplify the signal; it regenerates the signal. When it receives a weakened or corrupted signal, it creates a copy, bit for bit, at the original strength.

A repeater is a regenerator, not an amplifier.

The location of a repeater on a link is vital. A repeater must be placed so that a signal reaches it before any noise changes the meaning of any of its bits. A little noise can alter the precision of a bit's voltage without destroying its identity (see Figure 15.3). If the corrupted bit travels much farther, however, accumulated noise can change its meaning completely. At that point, the original voltage is not recoverable, and the error needs to be corrected. A repeater placed on the line before the legibility of the signal becomes lost can still read the signal well enough to determine the intended voltages and replicate them in their original form.

Figure 15.3 *Function of a repeater*



a. Right-to-teft transmission.

b. Left-to·right transmission.

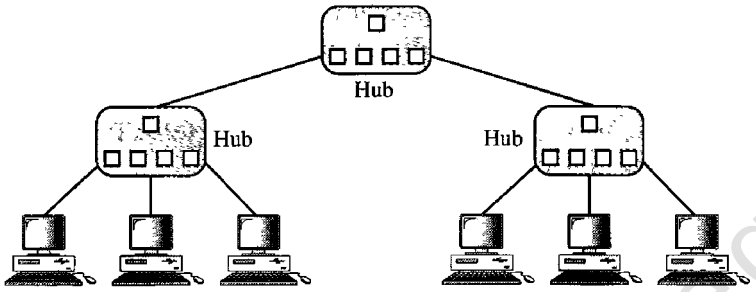
Active Hubs

An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology. We have seen examples of hubs in some Ethernet implementations (lOBase-T, for example). However, hubs can also be used to create multiple levels of hierarchy, as shown in Figure 15.4. The hierarchical use of hubs removes the length limitation of 10Base-T (100 m).

Bridges

A bridge operates in both the physical and the data link layer. As a physical layer device, it regenerates the signal it receives. As a data link layer device, the bridge can check the physical (MAC) addresses (source and destination) contained in the frame.

Figure 15.4 *A hierarchy of hubs*



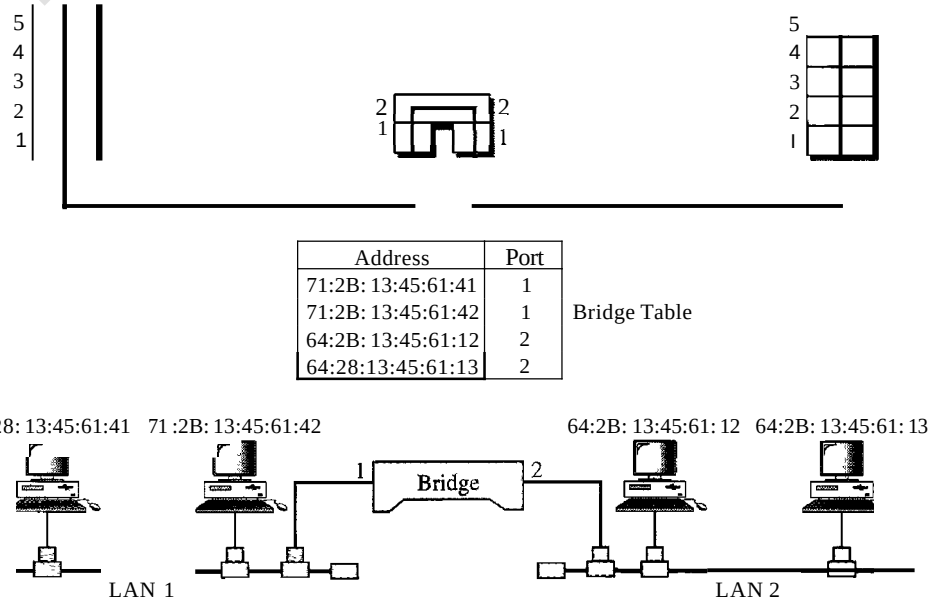
Filtering

One may ask, What is the difference in functionality between a bridge and a repeater? A bridge has filtering capability. It can check the destination address of a frame and decide if the frame should be forwarded or dropped. If the frame is to be forwarded, the decision must specify the port. A bridge has a table that maps addresses to ports.

A bridge has a table nsed in filtering decisions.

Let us give an example. In Figure 15.5, two LANs are connected by a bridge. If a frame destined for station 712B13456142 arrives at port 1, the bridge consults its table to find the departing port. According to its table, frames for 7l2B13456142 leave through port 1; therefore, there is no need for forwarding, and the frame is dropped. On the other hand, if a frame for 712B13456141 arrives at port 2, the departing port is port 1

Figure 15.5 *A bridge connecting two LANs*



and the frame is forwarded. In the first case, LAN 2 remains free of traffic; in the second case, both LANs have traffic. In our example, we show a two-port bridge; in reality a bridge usually has more ports.

Note also that a bridge does not change the physical addresses contained in the frame.

A bridge does not change the physical (MAC) addresses in a frame.

Transparent Bridges

A transparent bridge is a bridge in which the stations are completely unaware of the bridge's existence. If a bridge is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent bridges must meet three criteria:

1. Frames must be forwarded from one station to another.
2. The forwarding table is automatically made by learning frame movements in the network.
3. Loops in the system must be prevented.

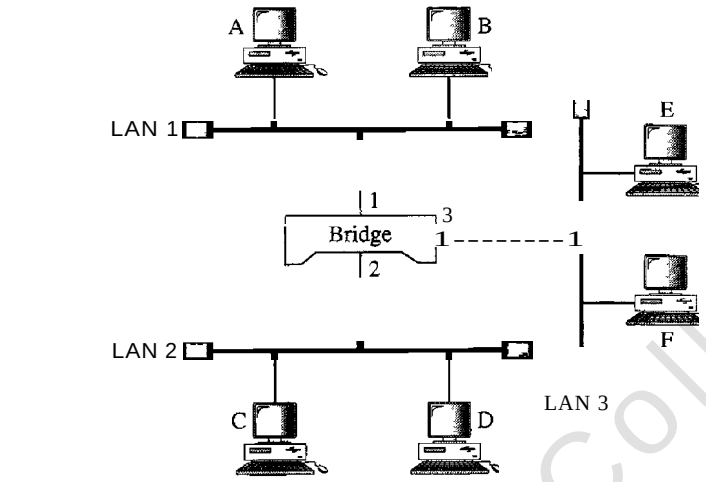
Forwarding A transparent bridge must correctly forward the frames, as discussed in the previous section.

Learning The earliest bridges had forwarding tables that were static. The systems administrator would manually enter each table entry during bridge setup. Although the process was simple, it was not practical. If a station was added or deleted, the table had to be modified manually. The same was true if a station's MAC address changed, which is not a rare event. For example, putting in a new network card means a new MAC address.

A better solution to the static table is a dynamic table that maps addresses to ports automatically. To make a table dynamic, we need a bridge that gradually learns from the frame movements. To do this, the bridge inspects both the destination and the source addresses. The destination address is used for the forwarding decision (table lookup); the source address is used for adding entries to the table and for updating purposes. Let us elaborate on this process by using Figure 15.6.

1. When station A sends a frame to station D, the bridge does not have an entry for either D or A. The frame goes out from all three ports; the frame floods the network. However, by looking at the source address, the bridge learns that station A must be located on the LAN connected to port 1. This means that frames destined for A, in the future, must be sent out through port 1. The bridge adds this entry to its table. The table has its first entry now.
2. When station E sends a frame to station A, the bridge has an entry for A, so it forwards the frame only to port 1. There is no flooding. In addition, it uses the source address of the frame, E, to add a second entry to the table.
3. When station B sends a frame to C, the bridge has no entry for C, so once again it floods the network and adds one more entry to the table.
4. The process of learning continues as the bridge forwards frames.

Figure 15.6 *A learning bridge and the process of learning*



Address	Port

a. Original

Address	Port
A	1

b. After A sends a frame to D

Address	Port
A	1
E	3

c. After E sends a frame to A

Address	Port
A	1
E	3
B	1

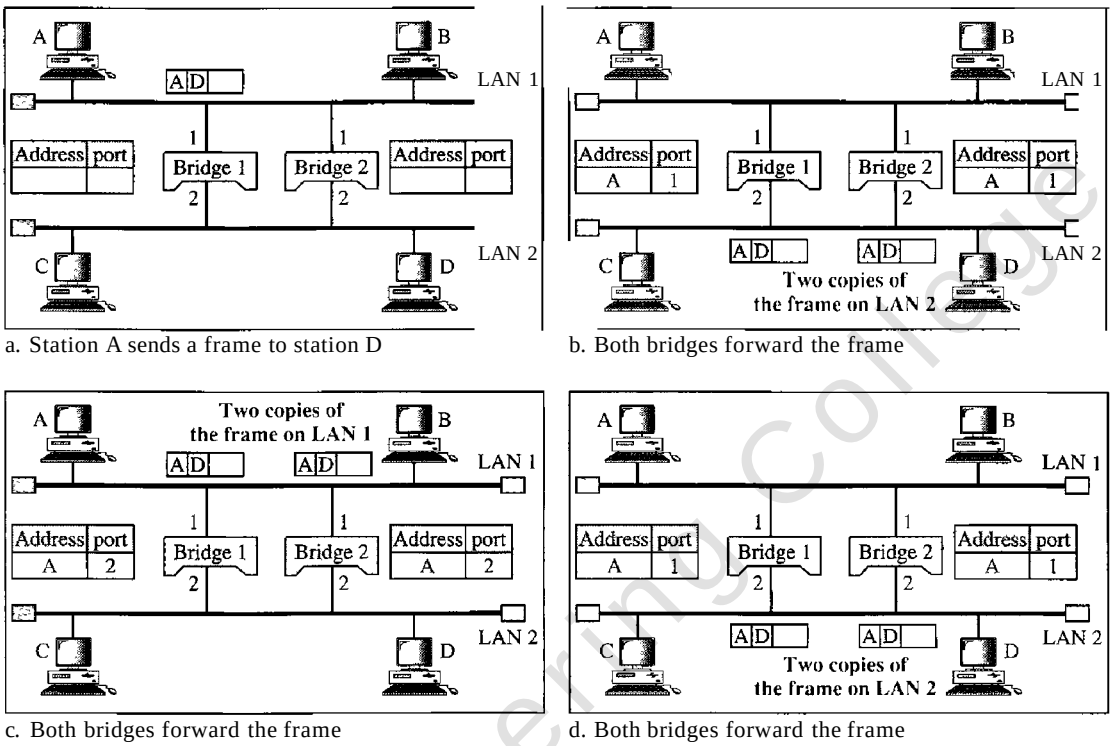
d. After B sends a frame to C

Loop Problem Transparent bridges work fine as long as there are no redundant bridges in the system. Systems administrators, however, like to have redundant bridges (more than one bridge between a pair of LANs) to make the system more reliable. If a bridge fails, another bridge takes over until the failed one is repaired or replaced. Redundancy can create loops in the system, which is very undesirable. Figure 15.7 shows a very simple example of a loop created in a system with two LANs connected by two bridges.

1. Station A sends a frame to station D. The tables of both bridges are empty. Both forward the frame and update their tables based on the source address A.
2. Now there are two copies of the frame on LAN 2. The copy sent out by bridge 1 is received by bridge 2, which does not have any information about the destination address D; it floods the bridge. The copy sent out by bridge 2 is received by bridge 1 and is sent out for lack of information about D. Note that each frame is handled separately because bridges, as two nodes on a network sharing the medium, use an access method such as CSMA/CD. The tables of both bridges are updated, but still there is no information for destination D.
3. Now there are two copies of the frame on LAN 1. Step 2 is repeated, and both copies flood the network.
4. The process continues on and on. Note that bridges are also repeaters and regenerate frames. So in each iteration, there are newly generated fresh copies of the frames.

To solve the looping problem, the IEEE specification requires that bridges use the spanning tree algorithm to create a loopless topology.

Figure 15.7 *Loop problem in a learning bridge*



a. Station A sends a frame to station D

b. Both bridges forward the frame

c. Both bridges forward the frame

d. Both bridges forward the frame

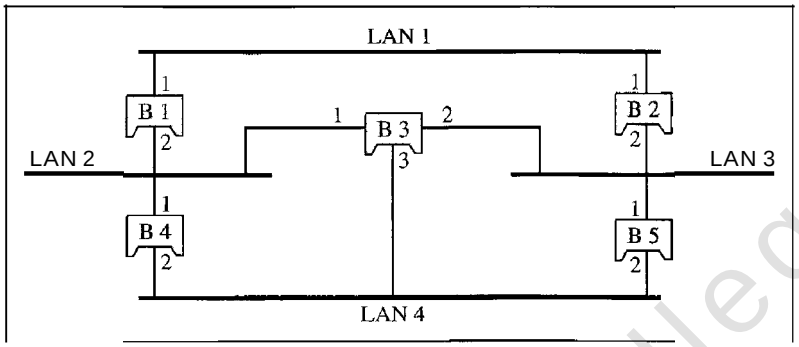
Spanning Tree

In graph theory, a **spanning tree** is a graph in which there is no loop. In a bridged LAN, this means creating a topology in which each LAN can be reached from any other LAN through one path only (no loop). We cannot change the physical topology of the system because of physical connections between cables and bridges, but we can create a logical topology that overlays the physical one. Figure 15.8 shows a system with four LANs and five bridges. We have shown the physical system and its representation in graph theory. Although some textbooks represent the LANs as nodes and the bridges as the connecting arcs, we have shown both LANs and bridges as nodes. The connecting arcs show the connection of a LAN to a bridge and vice versa. To find the spanning tree, we need to assign a cost (metric) to each arc. The interpretation of the cost is left up to the systems administrator. It may be the path with minimum hops (nodes), the path with minimum delay, or the path with maximum bandwidth. If two ports have the same shortest value, the systems administrator just chooses one. We have chosen the minimum hops. However, as we will see in Chapter 22, the hop count is normally 1 from a bridge to the LAN and 0 in the reverse direction.

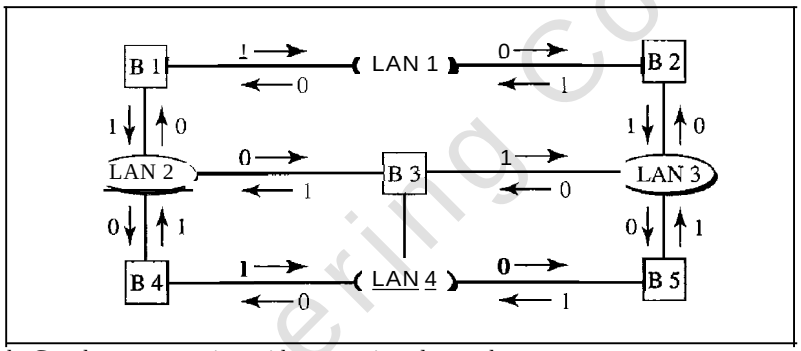
The process to find the spanning tree involves three steps:

1. Every bridge has a built-in ID (normally the serial number, which is unique). Each bridge broadcasts this ID so that all bridges know which one has the smallest ID. The bridge with the smallest ID is selected as the *root* bridge (root of the tree). We assume that bridge B1 has the smallest ID. It is, therefore, selected as the root bridge.

Figure 15.8 *A system of connected LANs and its graph representation*



a. Actual system



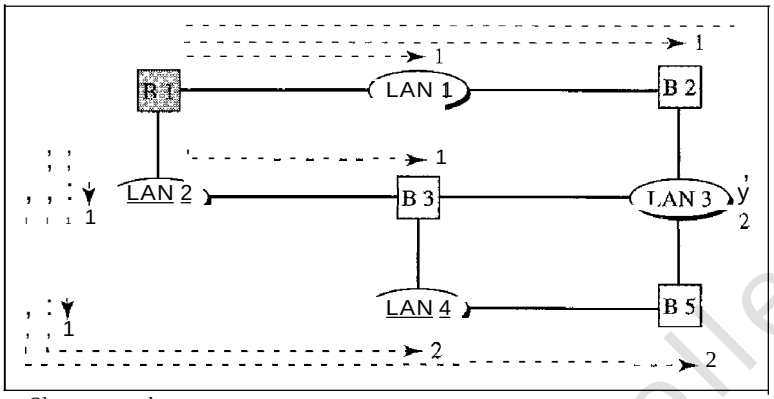
b. Graph representation with cost assigned to each arc

2. The algorithm tries to find the shortest path (a path with the shortest cost) from the root bridge to every other bridge or LAN. The shortest path can be found by examining the total cost from the root bridge to the destination. Figure 15.9 shows the shortest paths.
3. The combination of the shortest paths creates the shortest tree, which is also shown in Figure 15.9.
4. Based on the spanning tree, we mark the ports that are part of the spanning tree, the forwarding ports, which forward a frame that the bridge receives. We also mark those ports that are not part of the spanning tree, the blocking ports, which block the frames received by the bridge. Figure 15.10 shows the physical systems of LANs with forwarding points (solid lines) and blocking ports (broken lines).

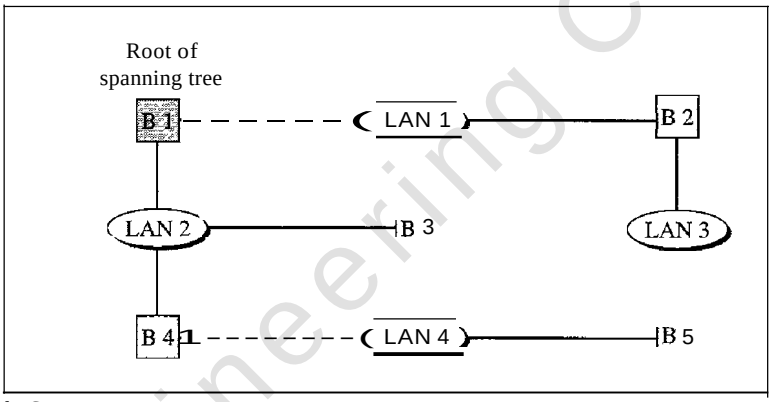
Note that there is only one single path from any LAN to any other LAN in the spanning tree system. This means there is only one single path from one LAN to any other LAN. No loops are created. You can prove to yourself that there is only one path from LAN 1 to LAN 2, LAN 3, or LAN 4. Similarly, there is only one path from LAN 2 to LAN 1, LAN 3, and LAN 4. The same is true for LAN 3 and LAN 4.

Dynamic Algorithm We have described the spanning tree algorithm as though it required manual entries. This is not true. Each bridge is equipped with a software package that carries out this process dynamically. The bridges send special messages to one another, called bridge protocol data units (BPDUs), to update the spanning tree. The spanning tree is updated when there is a change in the system such as a failure of a bridge or an addition or deletion of bridges.

Figure 15.9 *Finding the shortest paths and the spanning tree in a system of bridges*

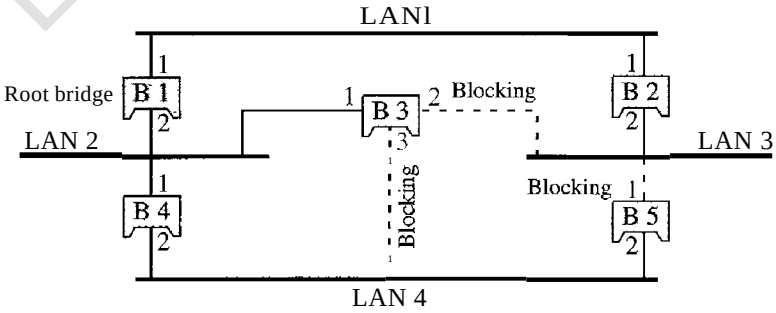


a. Shortest paths



b. Spanning tree

Figure 15.10 *Forwarding and blocking ports after using spanning tree algorithm*



Ports 2 and 3 of bridge B3 are blocking ports (no frame is sent out of these ports).
Port 1 of bridge B5 is also a blocking port (no frame is sent out of this port).

Source Routing Bridges

Another way to prevent loops in a system with redundant bridges is to use source **routing** bridges. A transparent bridge's duties include filtering frames, forwarding, and blocking. In a system that has source routing bridges, these duties are performed by the source station and, to some extent, the destination station.

In source routing, a sending station defines the bridges that the frame must visit. The addresses of these bridges are included in the frame. In other words, the frame contains not only the source and destination addresses, but also the addresses of all bridges to be visited.

The source gets these bridge addresses through the exchange of special frames with the destination prior to sending the data frame.

Source routing bridges were designed by IEEE to be used with Token Ring LANs. These LANs are not very common today.

Bridges Connecting Different LANs

Theoretically a bridge should be able to connect LANs using different protocols at the data link layer, such as an Ethernet LAN to a wireless LAN. However, there are many issues to be considered:

- **Frame format.** Each LAN type has its own frame format (compare an Ethernet frame with a wireless LAN frame).
- **Maximum data** size. If an incoming frame's size is too large for the destination LAN, the data must be fragmented into several frames. The data then need to be reassembled at the destination. However, no protocol at the data link layer allows the fragmentation and reassembly of frames. We will see in Chapter 19 that this is allowed in the network layer. The bridge must therefore discard any frames too large for its system.
- **Data rate.** Each LAN type has its own data rate. (Compare the 10-Mbps data rate of an Ethernet with the I-Mbps data rate of a wireless LAN.) The bridge must buffer the frame to compensate for this difference.
- **Bit order.** Each LAN type has its own strategy in the sending of bits. Some send the most significant bit in a byte first; others send the least significant bit first.
- **Security.** Some LANs, such as wireless LANs, implement security measures in the data link layer. Other LANs, such as Ethernet, do not. Security often involves encryption (see Chapter 30). When a bridge receives a frame from a wireless LAN, it needs to decrypt the message before forwarding it to an Ethernet LAN.
- **Multimedia support.** Some LANs support multimedia and the quality of services needed for this type of communication; others do not.

Two-Layer Switches

When we use the term *switch,* we must be careful because a switch can mean two different things. We must clarify the term by adding the level at which the device operates. We can have a two-layer switch or a three-layer switch. A **three-layer switch** is used at the network layer; it is a kind of router. The **two-layer switch** performs at the physical and data link layers.

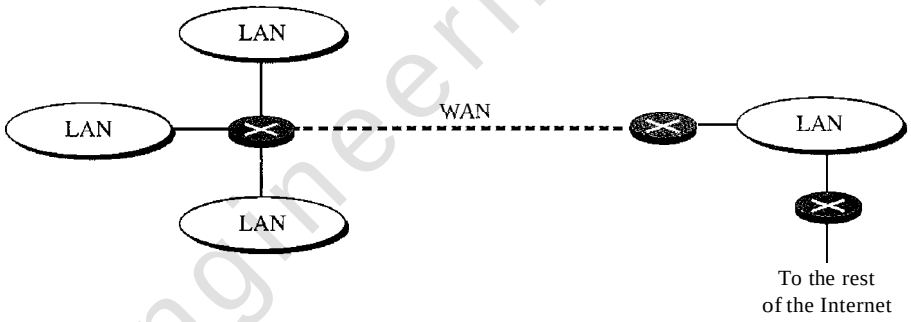
A two-layer switch is a bridge, a bridge with many ports and a design that allows better (faster) performance. A bridge with a few ports can connect a few LANs together. A bridge with many ports may be able to allocate a unique port to each station, with each station on its own independent entity. This means no competing traffic (no collision, as we saw in Ethernet).

A two-layer switch, as a bridge does, makes a filtering decision based on the MAC address of the frame it received. However, a two-layer switch can be more sophisticated. It can have a buffer to hold the frames for processing. It can have a switching factor that forwards the frames faster. Some new two-layer switches, called *cut-through* switches, have been designed to forward the frame as soon as they check the MAC addresses in the header of the frame.

Routers

A router is a three-layer device that routes packets based on their logical addresses (host-to-host addressing). A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route. The routing tables are normally dynamic and are updated using routing protocols. We discuss routers and routing in greater detail in Chapters 19 and 21. Figure 15.11 shows a part of the Internet that uses routers to connect LANs and WANs.

Figure 15.11 *Routers connecting independent LANs and WANs*



Three-Layer Switches

A three-layer switch is a router, but a faster and more sophisticated. The switching fabric in a three-layer switch allows faster table lookup and forwarding. In this book, we use the terms *router* and *three-layer switch* interchangeably.

Gateway

Although some textbooks use the terms *gateway* and *router* interchangeably, most of the literature distinguishes between the two. A gateway is normally a computer that operates in all five layers of the Internet or seven layers of OSI model. A gateway takes an application message, reads it, and interprets it. This means that it can be used as a connecting device between two internetworks that use different models. For example, a network designed to use the OSI model can be connected to another network using the Internet model. The gateway connecting the two systems can take a frame as it arrives from the first system, move it up to the OSI application layer, and remove the message.

Gateways can provide security. In Chapter 32, we learn that the gateway is used to filter unwanted application-layer messages.

15.2 BACKBONE NETWORKS

Some connecting devices discussed in this chapter can be used to connect LANs in a backbone network. A backbone network allows several LANs to be connected. In a backbone network, no station is directly connected to the backbone; the stations are part of a LAN, and the backbone connects the LANs. The backbone is itself a LAN that uses a LAN protocol such as Ethernet; each connection to the backbone is itself another LAN.

Although many different architectures can be used for a backbone, we discuss only the two most common: the bus and the star.

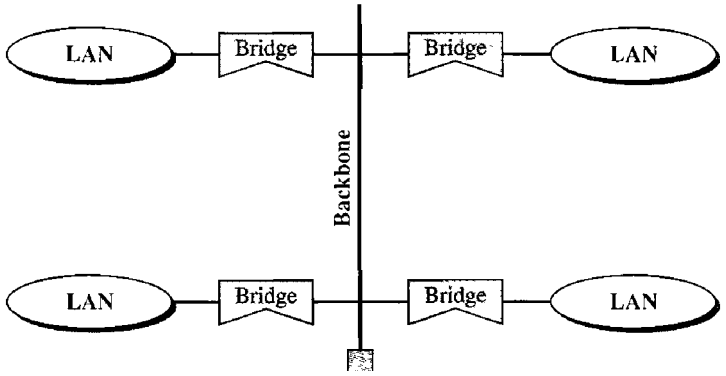
Bus Backbone

In a bus backbone, the topology of the backbone is a bus. The backbone itself can use one of the protocols that support a bus topology such as lOBase5 or lOBase2.

> In a bus backbone, the topology of the backbone is a bus.

Bus backbones are normally used as a distribution backbone to connect different buildings in an organization. Each building can comprise either a single LAN or another backbone (normally a star backbone). A good example of a bus backbone is one that connects single- or multiple-floor buildings on a campus. Each single-floor building usually has a single LAN. Each multiple-floor building has a backbone (usually a star) that connects each LAN on a floor. A bus backbone can interconnect these LANs and backbones. Figure 15.12 shows an example of a bridge-based backbone with four LANs.

Figure 15.12 *Bus backbone*



In Figure 15.12, if a station in a LAN needs to send a frame to another station in the same LAN, the corresponding bridge blocks the frame; the frame never reaches the backbone. However, if a station needs to send a frame to a station in another LAN, the bridge passes the frame to the backbone, which is received by the appropriate bridge and is delivered to the destination LAN. Each bridge connected to the backbone has a table that shows the stations on the LAN side of the bridge. The blocking or delivery of a frame is based on the contents of this table.

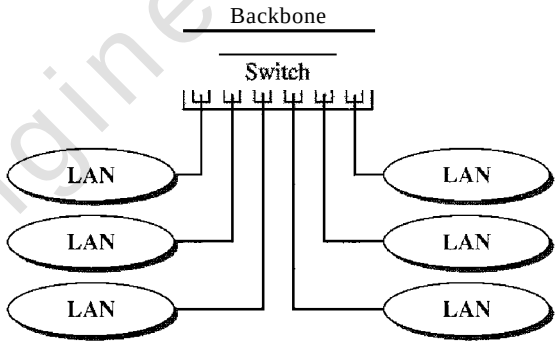
Star Backbone

In a star backbone, sometimes called a collapsed or switched backbone, the topology of the backbone is a star. In this configuration, the backbone is just one switch (that is why it is called, erroneously, a collapsed backbone) that connects the LANs.

In a star backbone, the topology of the backbone is a star; the backbone is just one switch.

Figure 15.13 shows a star backbone. Note that, in this configuration, the switch does the job of the backbone and at the same time connects the LANs.

Figure 15.13 *Star backbone*



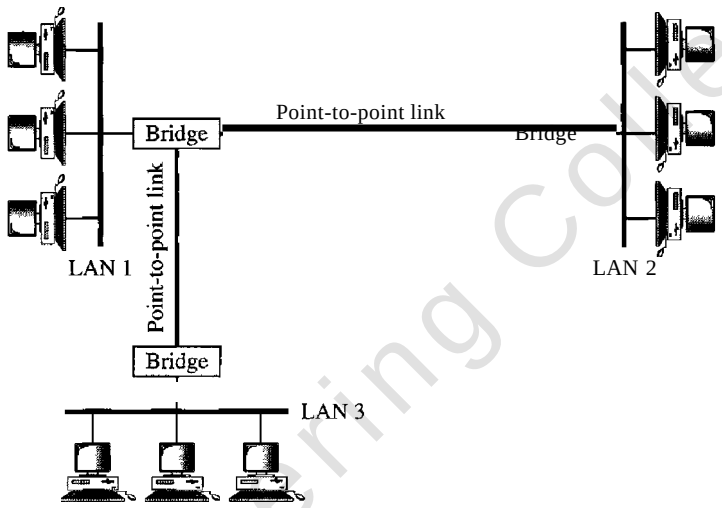
Star backbones are mostly used as a distribution backbone inside a building. In a multifloor building, we usually find one LAN that serves each particular floor. A star backbone connects these LANs. The backbone network, which is just a switch, can be installed in the basement or the first floor, and separate cables can run from the switch to each LAN. If the individual LANs have a physical star topology, either the hubs (or switches) can be installed in a closet on the corresponding floor, or all can be installed close to the switch. We often find a rack or chassis in the basement where the backbone switch and all hubs or switches are installed.

Connecting Remote LANs

Another common application for a backbone network is to connect remote LANs. This type of backbone network is useful when a company has several offices with LANs and needs to connect them. The connection can be done through bridges,

sometimes called remote bridges. The bridges act as connecting devices connecting LANs and point-to-point networks, such as leased telephone lines or ADSL lines. The point-to-point network in this case is considered a LAN without stations. The point-to-point link can use a protocol such as PPP. Figure 15.14 shows a backbone connecting remote LANs.

Figure 15.14 *Connecting remote IANs with bridges*



A point-to-point link acts as a LAN in a remote backbone connected by remote bridges.

15.3 VIRTUAL LANs

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a virtual local area network (VLAN) as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 15.15 shows a switched LAN in an engineering firm in which 10 stations are grouped into three LANs that are connected by a switch. The first four engineers work together as the first group, the next three engineers work together as the second group, and the last three engineers work together as the third group. The LAN is configured to allow this arrangement.

But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group? The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

Figure 15.15 *A switch connecting three LANs*

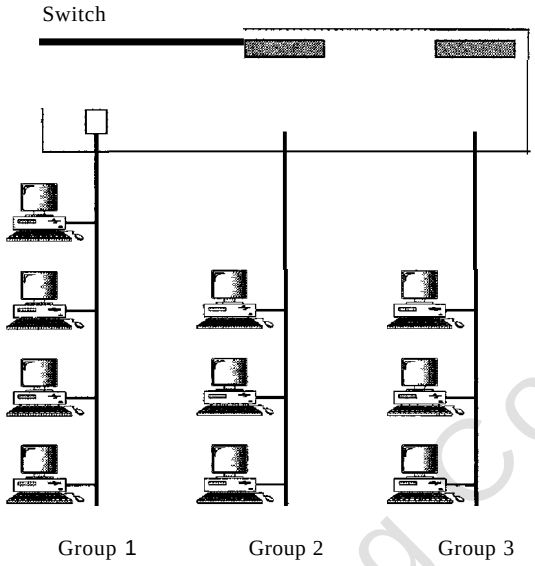
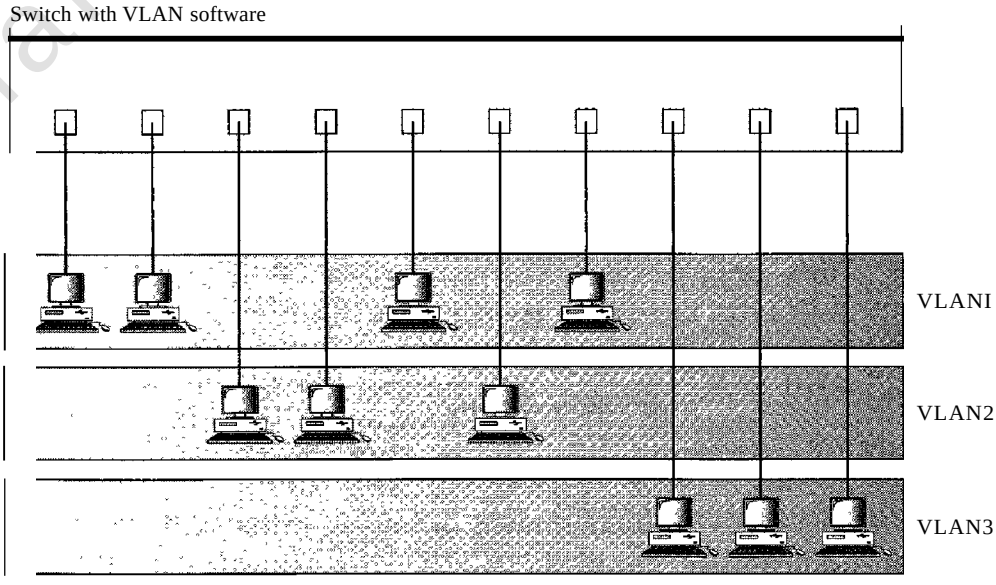


Figure 15.16 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs called VLANs. Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN. All members belonging to a VLAN can receive broadcast messages sent to that particular VLAN.

Figure 15.16 *A switch using VLAN software*

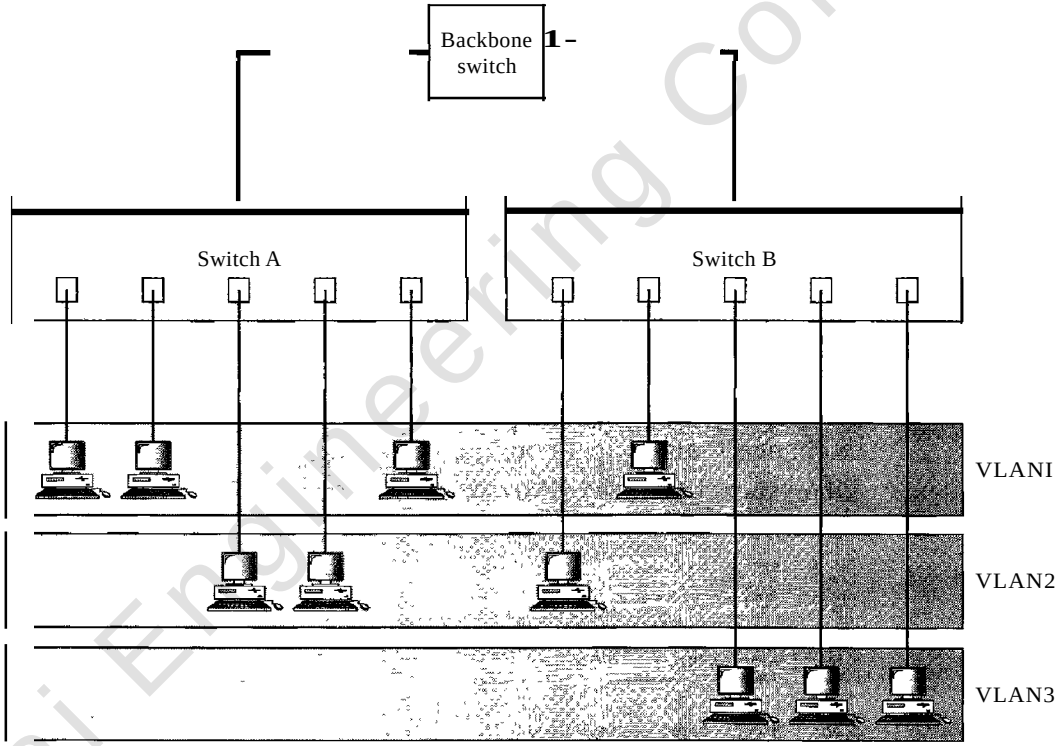


This means if a station moves from VLAN 1 to VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 15.17 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

Figure 15.17 *Two switches in a backbone using VLAN software*



This is a good configuration for a company with two separate buildings. Each building can have its own switched LAN connected by a backbone. People in the first building and people in the second building can be in the same work group even though they are connected to different physical LANs.

From these three examples, we can define a VLAN characteristic:

VLANs create broadcast domains.

VLANs group stations belonging to one or more physical LANs into broadcast domains. The stations in a VLAN communicate with one another as though they belonged to a physical segment.

Membership

What characteristic can be used to group stations in a VLAN? Vendors use different characteristics such as port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of two or more of these.

Port Numbers

Some VLAN vendors use switch port numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1; stations connecting to ports 4, 10, and 12 belong to VLAN 2; and so on.

MAC Addresses

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E21342A12334 and F2A123BCD341 belong to VLAN 1.

***IP** Addresses*

Some VLAN vendors use the 32-bit IP address (see Chapter 19) as a membership characteristic. For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

*Multicast **IP** Addresses*

Some VLAN vendors use the multicast IP address (see Chapter 19) as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the data link layer.

Combination

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

Configuration

How are the stations grouped into different VLANs? Stations are configured in one of three ways: manual, semiautomatic, and automatic.

Manual Configuration

In a manual configuration, the network administrator uses the VLAN software to manually assign the stations into different VLANs at setup. Later migration from one VLAN to another is also done manually. Note that this is not a physical configuration; it is a logical configuration. The term *manually* here means that the administrator types the port numbers, the IP addresses, or other characteristics, using the VLAN software.

Automatic Configuration

In an automatic configuration, the stations are automatically connected or disconnected from a VLAN using criteria defined by the administrator. For example, the administrator can define the project number as the criterion for being a member of a group. When a user changes the project, he or she automatically migrates to a new VLAN.

Semiautomatic Configuration

A semiautomatic configuration is somewhere between a manual configuration and an automatic configuration. Usually, the initializing is done manually, with migrations done automatically.

Communication Between Switches

In a multiswitched backbone, each switch must know not only which station belongs to which VLAN, but also the membership of stations connected to other switches. For example, in Figure 15.17, switch A must know the membership status of stations connected to switch B, and switch B must know the same about switch A. Three methods have been devised for this purpose: table maintenance, frame tagging, and time-division multiplexing.

Table Maintenance

In this method, when a station sends a broadcast frame to its group members, the switch creates an entry in a table and records station membership. The switches send their tables to one another periodically for updating.

Frame Tagging

In this method, when a frame is traveling between switches, an extra header is added to the MAC frame to define the destination VLAN. The frame tag is used by the receiving switches to determine the VLANs to be receiving the broadcast message.

Time-Division Multiplexing (TDM)

In this method, the connection (trunk) between switches is divided into timeshared channels (see TDM in Chapter 6). For example, if the total number of VLANs in a backbone is five, each trunk is divided into five channels. The traffic destined for VLAN 1 travels in channell, the traffic destined for VLAN 2 travels in channel 2, and so on. The receiving switch determines the destination VLAN by checking the channel from which the frame arrived.

IEEE Standard

In 1996, the IEEE 802.1 subcommittee passed a standard called 802.1Q that defines the format for frame tagging. The standard also defines the format to be used in multiswitched backbones and enables the use of multivendor equipment in VLANs. IEEE 802.1Q has opened the way for further standardization in other issues related to VLANs. Most vendors have already accepted the standard.

Advantages

There are several advantages to using VLANs.

Cost and Time Reduction

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

Creating Virtual Work Groups

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

Security

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.

15.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

A book devoted to connecting devices is [PerOO]. Connecting devices and VLANs are discussed in Section 4.7 of [Tan03]. Switches, bridges, and hubs are discussed in [Sta03] and [Sta04].

Site

- IEEE 802 LAN/MAN Standards Committee

15.5 KEY TERMS

amplifier
blocking port
bridge
bus backbone
connecting device
filtering
forwarding port
hub
remote bridge
repeater
router
segment

source routing bridge
spanning tree
star backbone
three-layer switch
transparent bridge
two-layer switch
virtual local area network (VLAN)

15.6 SUMMARY

- A repeater is a connecting device that operates in the physical layer of the Internet model. A repeater regenerates a signal, connects segments of a LAN, and has no filtering capability.
- A bridge is a connecting device that operates in the physical and data link layers of the Internet model.
- A transparent bridge can forward and filter frames and automatically build its forwarding table.
- A bridge can use the spanning tree algorithm to create a loopless topology.
- A backbone LAN allows several LANs to be connected.
- A backbone is usually a bus or a star.
- A virtual local area network (VLAN) is configured by software, not by physical wiring.
- Membership in a VLAN can be based on port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of these features.
- VLANs are cost- and time-efficient, can reduce network traffic, and provide an extra measure of security.

15.7 PRACTICE SET

Review Questions

1. How is a repeater different from an amplifier?
2. What do we mean when we say that a bridge can filter traffic? Why is filtering important?
3. What is a transparent bridge?
4. How does a repeater extend the length of a LAN?
5. How is a hub related to a repeater?
6. What is the difference between a forwarding port and a blocking port?
7. What is the difference between a bus backbone and a star backbone?
8. How does a VLAN save a company time and money?
9. How does a VLAN provide extra security for a network?
10. How does a VLAN reduce network traffic?
11. What is the basis for membership in a VLAN?

Exercises

12. Complete the table in Figure 15.6 after each station has sent a packet to another station.
13. Find the spanning tree for the system in Figure 15.7.
14. Find the spanning tree for the system in Figure 15.8 if bridge B5 is removed.
15. Find the spanning tree for the system in Figure 15.8 if bridge B2 is removed.
16. Find the spanning tree for the system in Figure 15.8 if B5 is selected as the root bridge.
17. In Figure 15.6, we are using a bridge. Can we replace the bridge with a router? Explain the consequences.
18. A bridge uses a filtering table; a router uses a routing table. Can you explain the difference?
19. Create a system of three LANs with four bridges. The bridges (B1 to B4) connect the LANs as follows:
    a. B1 connects LAN 1 and LAN 2.
    b. B2 connects LAN 1 and LAN 3.
    c. B3 connects LAN 2 and LAN 3.
    d. B4 connects LAN 1, LAN 2, and LAN 3.

    Choose B1as the root bridge. Show the forwarding and blocking ports, after applying the spanning tree procedure.
20. Which one has more overhead, a bridge or a router? Explain your answer.
21. Which one has more overhead, a repeater or a bridge? Explain your answer.
22. Which one has more overhead, a router or a gateway? Explain your answer.

PART

4

Network Layer

Objectives

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination.

The network layer is responsible for the delivery of individual packets from the source to the destination host.

The network layer adds a header that includes the logical addresses of the sender and receiver to the packet corning from the upper layer. If a packet travels through the Internet, we need this addressing system to help distinguish the source and destination.

When independent networks or links are connected together to create an internetwork, routers or switches route packets to their final destination. One of the functions of the network layer is to provide a routing mechanism.

In Part 4 of the book, we first discuss logical addressing (referred to as IP addressing in the Internet). We then discuss the main as well as some auxiliary protocols that are responsible for controlling the delivery of a packet from its source to its destination.

Part 4 of the book is devoted to the network layer and the services provided by this layer.

Chapters

This part consists of four chapters: Chapters 19 to 22.

Chapter 19

Chapter 19 discusses logical or IP addressing. We first discuss the historical classful addressing. We then describe the new classless addressing designed to alleviate some problems inherent in classful addressing. The completely new addressing system, IPv6, which may become prevalent in the near future, is also discussed.

Chapter 20

Chapter 20 is devoted to the main protocol at the network layer that supervises and controls the delivery of packets from the source to destination. This protocol is called the Internet Protocol or IP.

Chapter 21

Chapter 21 is devoted to some auxiliary protocols defined at the network layer, that help the IP protocol do its job. These protocols perform address mapping (logical to physical or vice versa), error reporting, and facilitate multicast delivery.

Chapter 22

Delivery and routing of packets in the Internet is a very delicate and important issue. We devote Chapter 22 to this matter. We first discuss the mechanism of delivery and routing. We then briefly discuss some unicast and multicast routing protocols used in the Internet today.

CHAPTER 19

Network Layer: Logical Addressing

As we discussed in Chapter 2, communication at the network layer is host-to-host (computer-to-computer); a computer somewhere in the world needs to communicate with another computer somewhere else in the world. Usually, computers communicate through the Internet. The packet transmitted by the sending computer may pass through several LANs or WANs before reaching the destination computer.

For this level of communication, we need a global addressing scheme; we called this logical addressing in Chapter 2. Today, we use the term IP address to mean a logical address in the network layer of the TCP/IP protocol suite.

The Internet addresses are 32 bits in length; this gives us a maximum of 2^{32} addresses. These addresses are referred to as IPv4 (IP version 4) addresses or simply IP addresses if there is no confusion.

The need for more addresses, in addition to other concerns about the IP layer, motivated a new design of the IP layer called the new generation of IP or IPv6 (lP version 6). In this version, the Internet uses 128-bit addresses that give much greater flexibility in address allocation. These addresses are referred to as IPv6 (IP version 6) addresses.

In this chapter, we first discuss IPv4 addresses, which are currently being used in the Internet. We then discuss the IPv6 addresses, which may become dominant in the future.

19.1 IPv4 ADDRESSES

An **IPv4** address is a 32-bit address that *uniquely* and *universally* defines the connection of a device (for example, a computer or a router) to the Internet.

An IPv4 address is 32 bits long.

IPv4 addresses are unique. They are unique in the sense that each address defines one, and only one, connection to the Internet. Two devices on the Internet can never have the same address at the same time. We will see later that, by using some strategies, an address may be assigned to a device for a time period and then taken away and assigned to another device.

On the other hand, if a device operating at the network layer has *m* connections to the Internet, it needs to have *m* addresses. We will see later that a router is such a device.

The IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

The IPv4 addresses are unique and universal.

Address Space

A protocol such as IPv4 that defines addresses has an address space. An address space is the total number of addresses used by the protocol. If a protocol uses *N* bits to define an address, the address space is 2^N because each bit can have two different values (0 or 1) and *N* bits can have 2^N values.

IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than 4 billion). This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet. We will see shortly that the actual number is much less because of the restrictions imposed on the addresses.

The address space of IPv4 is 2^{32} or 4,294,967,296.

Notations

There are two prevalent notations to show an IPv4 address: binary notation and dotted-decimal notation.

Binary Notation

In binary notation, the IPv4 address is displayed as 32 bits. Each octet is often referred to as a byte. So it is common to hear an IPv4 address referred to as a 32-bit address or a 4-byte address. The following is an example of an IPv4 address in binary notation:

01110101 10010101 00011101 00000010

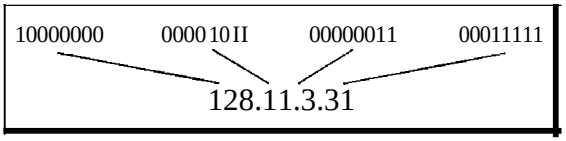
Dotted-Decimal Notation

To make the IPv4 address more compact and easier to read, Internet addresses are usually written in decimal form with a decimal point (dot) separating the bytes. The following is the **dotted-decimal** notation of the above address:

117.149.29.2

Figure 19.1 shows an IPv4 address in both binary and dotted-decimal notation. Note that because each byte (octet) is 8 bits, each number in dotted-decimal notation is a value ranging from 0 to 255.

Figure 19.1 *Dotted-decimal notation and binary notation for an IPv4 address*



Numbering systems are reviewed **in** Appendix B.

Example 19.1

Change the following IPv4 addresses from binary notation to dotted-decimal notation.

a. 10000001 00001011 00001011 11101111
b. 11000001 10000011 00011011 11111111

Solution

We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation.

a. 129.11.11.239
b. 193.131.27.255

Example 19.2

Change the following IPv4 addresses from dotted-decimal notation to binary notation.

a. 111.56.45.78
b. 221.34.7.82

Solution

We replace each decimal number with its binary equivalent (see Appendix B).

a. 01101111 00111000 00101101 01001110
b. 11011101 00100010 00000111 01010010

Example 19.3

Find the error, if any, in the following IPv4 addresses.

a. 111.56.045.78
b. 221.34.7.8.20
c. 75.45.301.14
d. 11100010.23.14.67

Solution

a. There must be no leading zero (045).
b. There can be no more than four numbers in an IPv4 address.
c. Each number needs to be less than or equal to 255 (301 is outside this range).
d. A mixture of binary notation and dotted-decimal notation is not allowed.

Classful Addressing

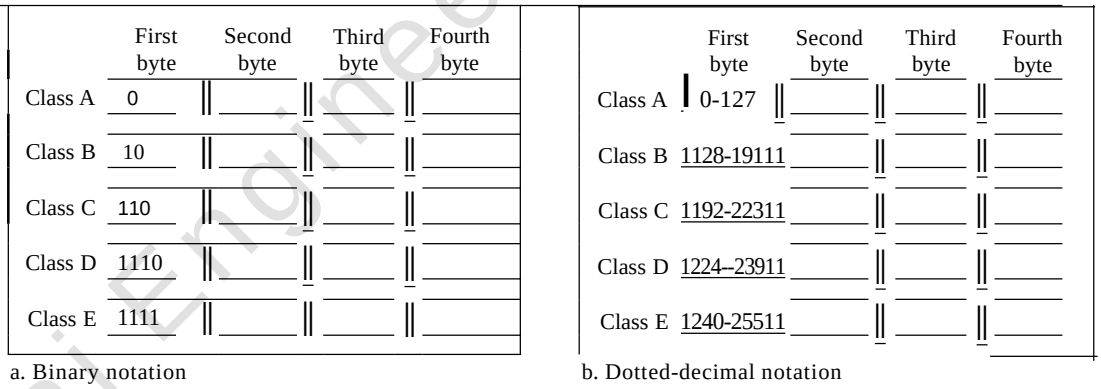
IPv4 addressing, at its inception, used the concept of classes. This architecture is called classful addressing. Although this scheme is becoming obsolete, we briefly discuss it here to show the rationale behind classless addressing.

In classful addressing, the address space is divided into five classes: A, B, C, D, and E. Each class occupies some part of the address space.

In classful addressing, the address space is divided into five classes: A, B, C, D, and E.

We can find the class of an address when given the address in binary notation or dotted-decimal notation. If the address is given in binary notation, the first few bits can immediately tell us the class of the address. If the address is given in decimal-dotted notation, the first byte defines the class. Both methods are shown in Figure 19.2.

Figure 19.2 *Finding the classes in binary and dotted-decimal notation*



Example 19.4

Find the class of each address.

a. 00000001 00001011 00001011 11101111
b. 11000001 10000011 00011011 11111111
c. 14.23.120.8
d. 252.5.15.111

Solution

a. The first bit is 0. This is a class A address.
b. The first 2 bits are 1; the third bit is 0. This is a class C address.
c. The first byte is 14 (between 0 and 127); the class is A.
d. The first byte is 252 (between 240 and 255); the class is E.

Classes and Blocks

One problem with classful addressing is that each class is divided into a fixed number of blocks with each block having a fixed size as shown in Table 19.1.

Table 19.1 *Number of blocks and block size in classful IPv4 addressing*

Class	*Number of Blocks*	*Block Size*	*Application*
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

Let us examine the table. Previously, when an organization requested a block of addresses, it was granted one in class A, B, or C. Class A addresses were designed for large organizations with a large number of attached hosts or routers. Class B addresses were designed for midsize organizations with tens of thousands of attached hosts or routers. Class C addresses were designed for small organizations with a small number of attached hosts or routers.

We can see the flaw in this design. A block in class A address is too large for almost any organization. This means most of the addresses in class A were wasted and were not used. A block in class B is also very large, probably too large for many of the organizations that received a class B block. A block in class C is probably too small for many organizations. Class D addresses were designed for multicasting as we will see in a later chapter. Each address in this class is used to define one group of hosts on the Internet. The Internet authorities wrongly predicted a need for 268,435,456 groups. This never happened and many addresses were wasted here too. And lastly, the class E addresses were reserved for future use; only a few were used, resulting in another waste of addresses.

In c1assfnl addressing, a large part of the available addresses were wasted.

Netid and Hostid

In classful addressing, an IP address in class A, B, or C is divided into netid and hostid. These parts are of varying lengths, depending on the class of the address. Figure 19.2 shows some netid and hostid bytes. The netid is in color, the hostid is in white. Note that the concept does not apply to classes D and E.

In class A, one byte defines the netid and three bytes define the hostid. In class B, two bytes define the netid and two bytes define the hostid. In class C, three bytes define the netid and one byte defines the hostid.

Mask

Although the length of the netid and hostid (in bits) is predetermined in classful addressing, we can also use a mask (also called the default mask), a 32-bit number made of

contiguous Is followed by contiguous as. The masks for classes A, B, and C are shown in Table 19.2. The concept does not apply to classes D and E.

Table 19.2 *Default masks for classful addressing*

Class	*Binary*	*Dotted-Decimal*	*CIDR*
A	11111111 00000000 00000000 00000000	255.0.0.0	*18*
B	11111111 11111111 00000000 00000000	255.255.0.0	*116*
C	11111111 11111111 11111111 00000000	255.255.255.0	*124*

The mask can help us to find the netid and the hostid. For example, the mask for a class A address has eight 1s, which means the first 8 bits of any address in class A define the netid; the next 24 bits define the hostid.

The last column of Table 19.2 shows the mask in the form *In* where *n* can be 8, 16, or 24 in classful addressing. This notation is also called slash notation or Classless Interdomain Routing (CIDR) notation. The notation is used in classless addressing, which we will discuss later. We introduce it here because it can also be applied to classful addressing. We will show later that classful addressing is a special case of classless addressing.

Subnetting

During the era of classful addressing, subnetting was introduced. If an organization was granted a large block in class A or B, it could divide the addresses into several contiguous groups and assign each group to smaller networks (called subnets) or, in rare cases, share part of the addresses with neighbors. Subnetting increases the number of Is in the mask, as we will see later when we discuss classless addressing.

Supernetting

The time came when most of the class A and class B addresses were depleted; however, there was still a huge demand for midsize blocks. The size of a class C block with a maximum number of 256 addresses did not satisfy the needs of most organizations. Even a midsize organization needed more addresses. One solution was supernetting. In supernetting, an organization can combine several class C blocks to create a larger range of addresses. In other words, several networks are combined to create a supernetwork or a supemet. An organization can apply for a set of class C blocks instead of just one. For example, an organization that needs 1000 addresses can be granted four contiguous class C blocks. The organization can then use these addresses to create one supernetwork. Supernetting decreases the number of Is in the mask. For example, if an organization is given four class C addresses, the mask changes from /24 to /22. We will see that classless addressing eliminated the need for supernetting.

Address Depletion

The flaws in classful addressing scheme combined with the fast growth of the Internet led to the near depletion of the available addresses. Yet the number of devices on the Internet is much less than the 2^{32} address space. We have run out of class A and B addresses, and

a class C block is too small for most midsize organizations. One solution that has alleviated the problem is the idea of classless addressing.

Classful addressing, which is almost obsolete, is replaced with classless addressing.

Classless Addressing

To overcome address depletion and give more organizations access to the Internet, classless addressing was designed and implemented. In this scheme, there are no classes, but the addresses are still granted in blocks.

Address Blocks

In classless addressing, when an entity, small or large, needs to be connected to the Internet, it is granted a block (range) of addresses. The size of the block (the number of addresses) varies based on the nature and size of the entity. For example, a household may be given only two addresses; a large organization may be given thousands of addresses. An ISP, as the Internet service provider, may be given thousands or hundreds of thousands based on the number of customers it may serve.

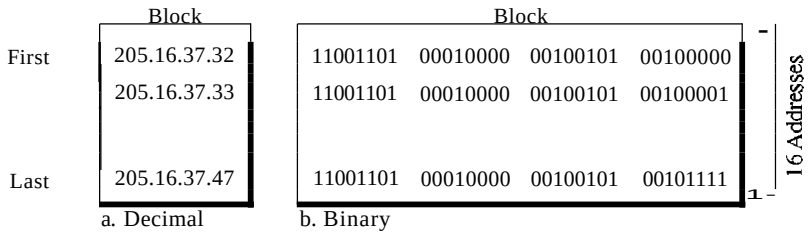
Restriction To simplify the handling of addresses, the Internet authorities impose three restrictions on classless address blocks:

1. The addresses in a block must be contiguous, one after another.
2. The number of addresses in a block must be a power of 2 (I, 2, 4, 8, ...).
3. The first address must be evenly divisible by the number of addresses.

Example 19.5

Figure 19.3 shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.

Figure 19.3 *A block of* 16 *addresses granted to a small organization*



We can see that the restrictions are applied to this block. The addresses are contiguous. The number of addresses is a power of 2 ($16 = 2^4$), and the first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in 215,024,210. In Appendix B, we show how to find the decimal value of an IP address.

Mask

A better way to define a block of addresses is to select any address in the block and the mask. As we discussed before, a mask is a 32-bit number in which the *n* leftmost bits are Is and the 32 - *n* rightmost bits are Os. However, in classless addressing the mask for a block can take any value from 0 to 32. It is very convenient to give just the value of *n* preceded by a slash (CIDR notation).

In 1Pv4 addressing, a block of addresses can be defined as
x.y.z.tln
in which x.y.z.t defines one of the addresses and the *In* defines the mask.

The address and the *In* notation completely define the whole block (the first address, the last address, and the number of addresses).

First Address The first address in the block can be found by setting the 32 - *n* rightmost bits in the binary notation of the address to Os.

The first address in the block can be found by setting the rightmost 32 - *n* bits to Os.

Example 19.6

A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

Solution

The binary representation of the given address is 11001101 00010000 00100101 00100 I 11. If we set 32 - 28 rightmost bits to 0, we get 11001101 000100000100101 0010000 or 205.16.37.32. This is actually the block shown in Figure 19.3.

Last Address The last address in the block can be found by setting the 32 - *n* rightmost bits in the binary notation of the address to Is.

The last address in the block can be found by setting the rightmost 32 - *n* bits to Is.

Example 19.7

Find the last address for the block in Example 19.6.

Solution

The binary representation of the given address is 11001101 00010000001001010010011 1. If we set 32 - 28 rightmost bits to 1, we get 11001101 00010000 0010010100101111 or 205.16.37.47. This is actually the block shown in Figure 19.3.

Number of Addresses The number of addresses in the block is the difference between the last and first address. It can easily be found using the formula 2^{32-n}.

The number of addresses in the block can be found by using the formula 2^{32-n}.

Example 19.8

Find the number of addresses in Example 19.6.

Solution

The value of n is 28, which means that number of addresses is 2^{32-28} or 16.

Example 19.9

Another way to find the first address, the last address, and the number of addresses is to represent the mask as a 32-bit binary (or 8-digit hexadecimal) number. This is particularly useful when we are writing a program to find these pieces of information. In Example 19.5 the /28 can be represented as 11111111 11111111 11111111 11110000 (twenty-eight Is and four Os). Find

a. The first address
b. The last address
c. The number of addresses

Solution

a. The first address can be found by ANDing the given addresses with the mask. ANDing here is done bit by bit. The result of ANDing 2 bits is 1 if both bits are Is; the result is 0 otherwise.

Address:	11001101	00010000	00100101	00100111
Mask:	11111111	11111111	11111111	11110000
First address:	11001101	00010000	00100101	00100000

b. The last address can be found by ORing the given addresses with the complement of the mask. ORing here is done bit by bit. The result of ORing 2 bits is 0 if both bits are Os; the result is 1 otherwise. The complement of a number is found by changing each 1 to 0 and each 0 to 1.

Address:	11001101	00010000	00100101	00100111
Mask complement:	00000000	00000000	00000000	00001111
Last address:	11001101	00010000	*00100101*	*00101111*

c. The number of addresses can be found by complementing the mask, interpreting it as a decimal number, and adding 1 to it.

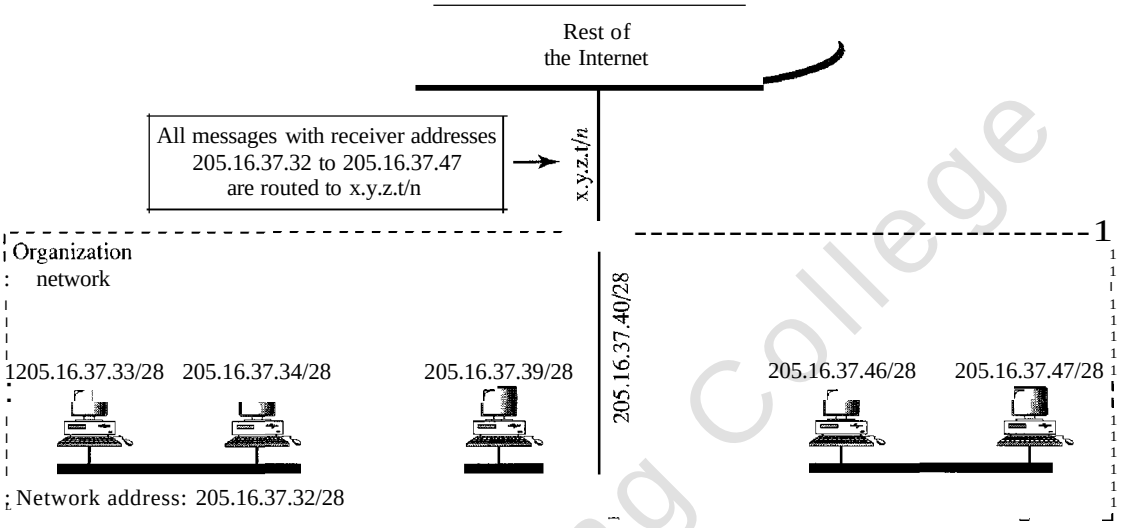
Mask complement:	000000000	00000000	00000000	00001111
Number of addresses:	15 + 1 = 16			

Network Addresses

A very important concept in IP addressing is the network address. When an organization is given a block of addresses, the organization is free to allocate the addresses to the devices that need to be connected to the Internet. The first address in the class, however, is normally (not always) treated as a special address. The first address is called the network address and defines the organization network. It defines the organization itself to the rest of the world. In a later chapter we will see that the first address is the one that is used by routers to direct the message sent to the organization from the outside.

Figure 19.4 shows an organization that is granted a 16-address block.

Figure 19.4 *A network configuration for the block 205.16.37.32/28*



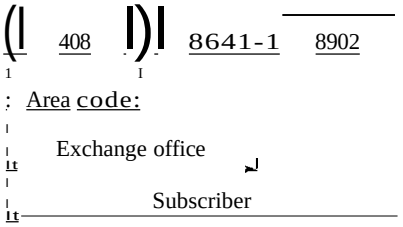
The organization network is connected to the Internet via a router. The router has two addresses. One belongs to the granted block; the other belongs to the network that is at the other side of the router. We call the second address *x.y.z.t/n* because we do not know anything about the network it is connected to at the other side. All messages destined for addresses in the organization block (205.16.37.32 to 205.16.37.47) are sent, directly or indirectly, to *x.y.z.t/n.* We say directly or indirectly because we do not know the structure of the network to which the other side of the router is connected.

The first address in a block is normally not assigned to any device; it is used as the network address that represents the organization to the rest of the world.

Hierarchy

IP addresses, like other addresses or identifiers we encounter these days, have levels of hierarchy. For example, a telephone network in North America has three levels of hierarchy. The leftmost three digits define the area code, the next three digits define the exchange, the last four digits define the connection of the local loop to the central office. Figure 19.5 shows the structure of a hierarchical telephone number.

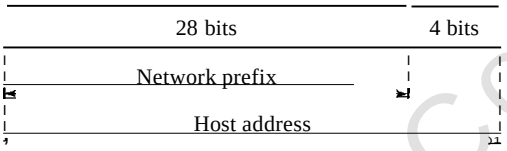
Figure 19.5 *Hierarchy in a telephone network in North America*



Two-Level Hierarchy: No Subnetting

An IP address can define only two levels of hierarchy when not subnetted. The *n* leftmost bits of the address *x.y.z.tJn* define the network (organization network); the 32 - *n* rightmost bits define the particular host (computer or router) to the network. The two common terms are prefix and suffix. The part of the address that defines the network is called the prefix; the part that defines the host is called the suffix. Figure 19.6 shows the hierarchical structure of an IPv4 address.

Figure 19.6 *Two levels of hierarchy in an IPv4 address*



The prefix is common to all addresses in the network; the suffix changes from one device to another.

Each address in the block can be considered as a two-level hierarchical structure:
the leftmost *n* bits (prefix) define the network;
the rightmost 32 - *n* bits define the host.

Three-Levels of Hierarchy: Subnetting

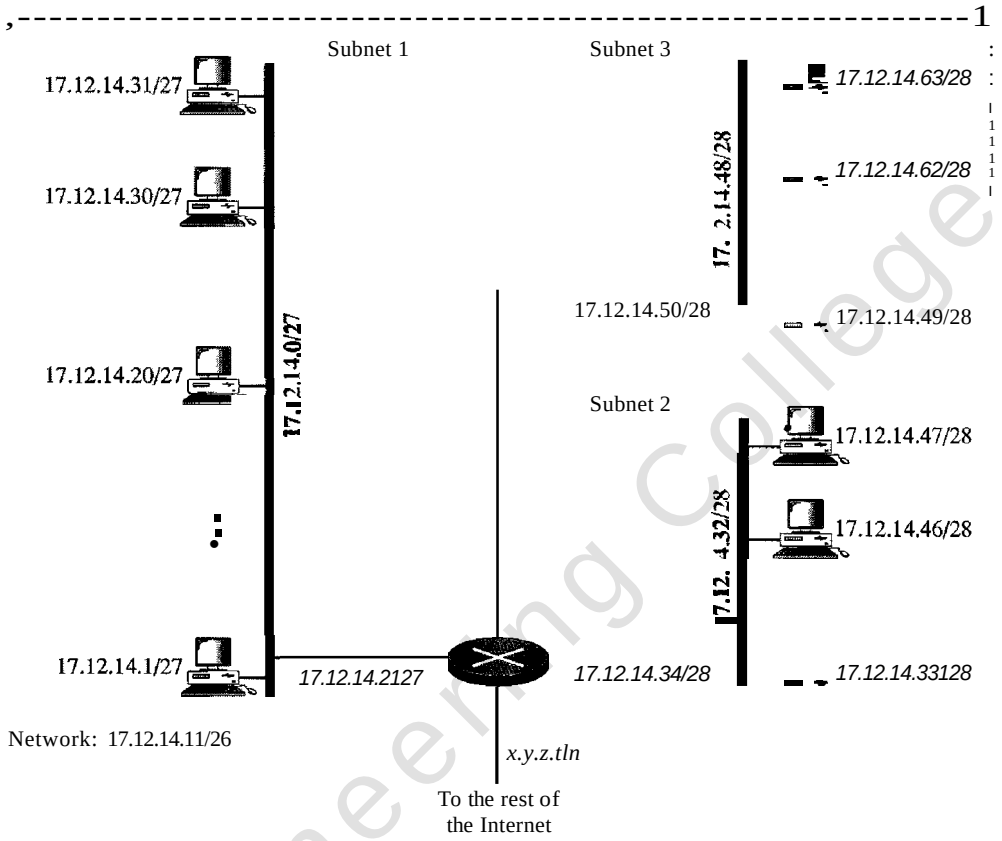
An organization that is granted a large block of addresses may want to create clusters of networks (called subnets) and divide the addresses between the different subnets. The rest of the world still sees the organization as one entity; however, internally there are several subnets. All messages are sent to the router address that connects the organization to the rest of the Internet; the router routes the message to the appropriate subnets. The organization, however, needs to create small subblocks of addresses, each assigned to specific subnets. The organization has its own mask; each subnet must also have its own.

As an example, suppose an organization is given the block 17.12.40.0/26, which contains 64 addresses. The organization has three offices and needs to divide the addresses into three subblocks of 32, 16, and 16 addresses. We can find the new masks by using the following arguments:

1. Suppose the mask for the first subnet is n1, then 2^{32-n1} must be 32, which means that n1 =27.
2. Suppose the mask for the second subnet is n2, then 2^{32-n2} must be 16, which means that n2 = 28.
3. Suppose the mask for the third subnet is n3, then 2^{32-n3} must be 16, which means that n3 =28.

This means that we have the masks 27, 28, 28 with the organization mask being 26. Figure 19.7 shows one configuration for the above scenario.

Figure 19.7 *Configuration and addresses in a subnetted network*



Let us check to see if we can find the subnet addresses from one of the addresses in the subnet.

a. In subnet 1, the address 17.12.14.29/27 can give us the subnet address if we use the mask /27 because

Host: 00010001 00001100 00001110 00011101
Mask: /27
Subnet: 00010001 00001100 00001110 00000000 .... (17.12.14.0)

b. In subnet 2, the address 17.12.14.45/28 can give us the subnet address if we use the mask /28 because

Host: 00010001 00001100 00001110 00101101
Mask: /28
Subnet: 00010001 00001100 00001110 00100000 .... (17.12.14.32)

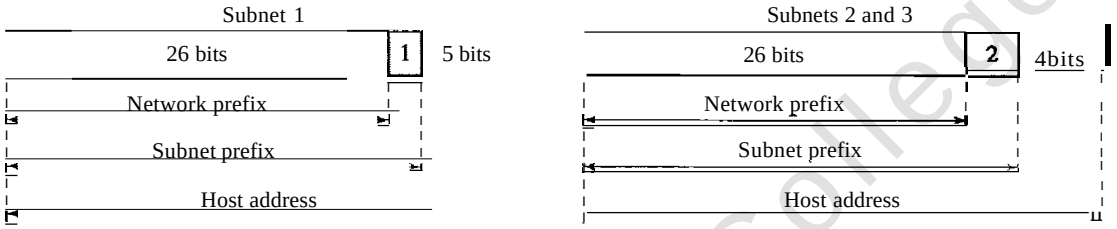
c. In subnet 3, the address 17.12.14.50/28 can give us the subnet address if we use the mask /28 because

Host: 00010001 00001100 00001110 00110010
Mask: /28
Subnet: 00010001 00001100 00001110 00110000 .... (17.12.14.48)

Note that applying the mask of the network, *126*, to any of the addresses gives us the network address 17.12.14.0/26. We leave this proof to the reader.

We can say that through subnetting, we have three levels of hierarchy. Note that in our example, the subnet prefix length can differ for the subnets as shown in Figure 19.8.

Figure 19.8 *Three-level hierarchy in an IPv4 address*



More Levels of Hierarchy

The structure of classless addressing does not restrict the number of hierarchical levels. An organization can divide the granted block of addresses into subblocks. Each subblock can in turn be divided into smaller subblocks. And so on. One example of this is seen in the ISPs. A national ISP can divide a granted large block into smaller blocks and assign each of them to a regional ISP. A regional ISP can divide the block received from the national ISP into smaller blocks and assign each one to a local ISP. A local ISP can divide the block received from the regional ISP into smaller blocks and assign each one to a different organization. Finally, an organization can divide the received block and make several subnets out of it.

Address Allocation

The next issue in classless addressing is address allocation. How are the blocks allocated? The ultimate responsibility of address allocation is given to a global authority called the *Internet Corporation for Assigned Names and Addresses* (ICANN). However, ICANN does not normally allocate addresses to individual organizations. It assigns a large block of addresses to an ISP. Each ISP, in turn, divides its assigned block into smaller subblocks and grants the subblocks to its customers. In other words, an ISP receives one large block to be distributed to its Internet users. This is called address aggregation: many blocks of addresses are aggregated in one block and granted to one ISP.

Example 19.10

An ISP is granted a block of addresses starting with 190.100.0.0/16 (65,536 addresses). The ISP needs to distribute these addresses to three groups of customers as follows:

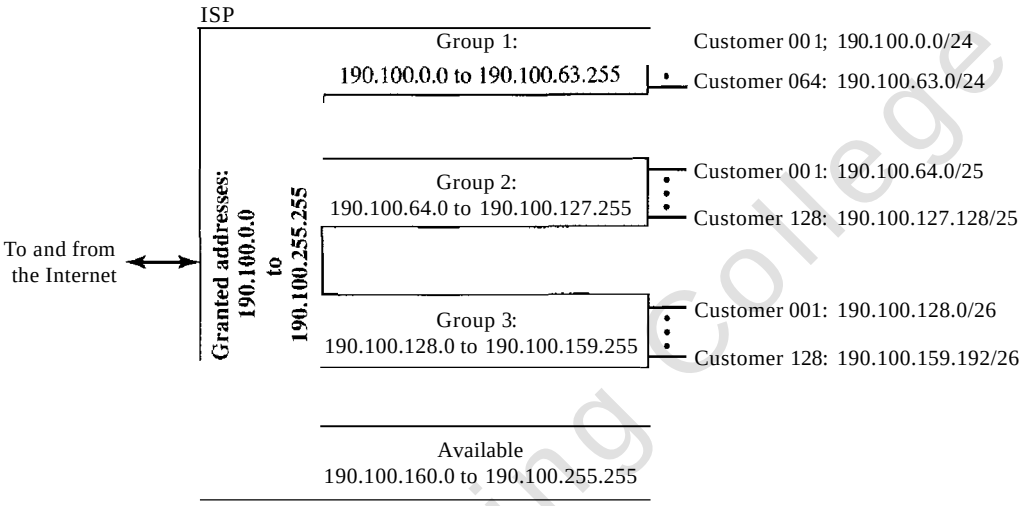
a. The first group has 64 customers; each needs 256 addresses.
b. The second group has 128 customers; each needs 128 addresses.
c. The third group has 128 customers; each needs 64 addresses.

Design the subblocks and find out how many addresses are still available after these allocations.

Solution

Figure 19.9 shows the situation.

Figure 19.9 *An example ofaddress allocation and distribution by an IS?*



1. **Group 1**

   For this group, each customer needs 256 addresses. This means that 8 (log2256) bits are needed to define each host. The prefix length is then 32 - 8 =24. The addresses are

   1st Customer: *190.100.0.0/24* *190.100.0.255/24*
   2nd Customer: *190.100.1.0/24* *190.100.1.255/24*

   64th Customer: *190.100.63.0/24* *190.100.63.255/24*
   Total =64 x 256 = 16,384

2. **Group2**

   For this group, each customer needs 128 addresses. This means that 7 (10g2 128) bits are needed to define each host. The prefix length is then 32 – 7 =25. The addresses are

   1st Customer: *190.100.64.0/25* *190.100.64.127/25*
   2nd Customer: *190.100.64.128/25* *190.100.64.255/25*

   128th Customer: *190.100.127.128/25* *190.100.127.255/25*
   Total =128 x 128 = 16,384

3. **Group3**

   For this group, each customer needs 64 addresses. This means that 6 (logz 64) bits are needed to each host. The prefix length is then 32 - 6 =26. The addresses are

1st Customer: *190.100.128.0/26* *190.100.128.63/26*
2nd Customer: *190.100.128.64/26* *190.100.128.127/26*

128th Customer: *190.100.159.192/26* *190.100.159.255/26*
Total = 128 x 64 = 8192

Number of granted addresses to the ISP: 65,536
Number of allocated addresses by the ISP: 40,960
Number of available addresses: 24,576

Network Address **Translation** (NAT)

The number of home users and small businesses that want to use the Internet is ever increasing. In the beginning, a user was connected to the Internet with a dial-up line, which means that she was connected for a specific period of time. An ISP with a block of addresses could dynamically assign an address to this user. An address was given to a user when it was needed. But the situation is different today. Home users and small businesses can be connected by an ADSL line or cable modem. In addition, many are not happy with one address; many have created small networks with several hosts and need an IP address for each host. With the shortage of addresses, this is a serious problem.

A quick solution to this problem is called network address translation (NAT). NAT enables a user to have a large set of addresses internally and one address, or a small set of addresses, externally. The traffic inside can use the large set; the traffic outside, the small set.

To separate the addresses used inside the home or business and the ones used for the Internet, the Internet authorities have reserved three sets of addresses as private addresses, shown in Table 19.3.

Table 19.3 *Addresses for private networks*

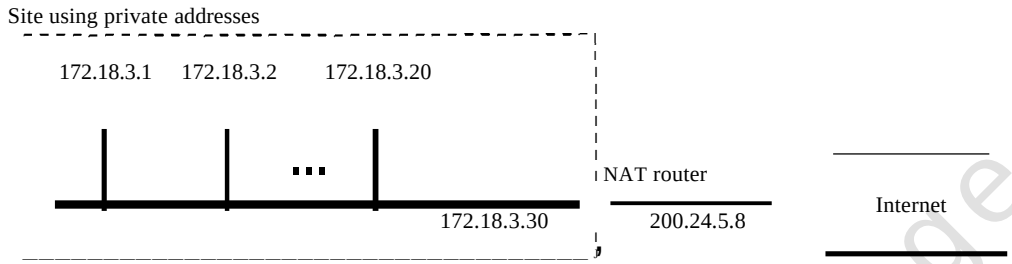
Range			*Total*
10.0.0.0	to	10.255.255.255	2^{24}
172.16.0.0	to	172.31.255.255	2^{20}
192.168.0.0	to	192.168.255.255	2^{16}

Any organization can use an address out of this set without permission from the Internet authorities. Everyone knows that these reserved addresses are for private networks. They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address.

The site must have only one single connection to the global Internet through a router that runs the NAT software. Figure 19.10 shows a simple implementation of NAT.

As Figure 19.10 shows, the private network uses private addresses. The router that connects the network to the global address uses one private address and one global address. The private network is transparent to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

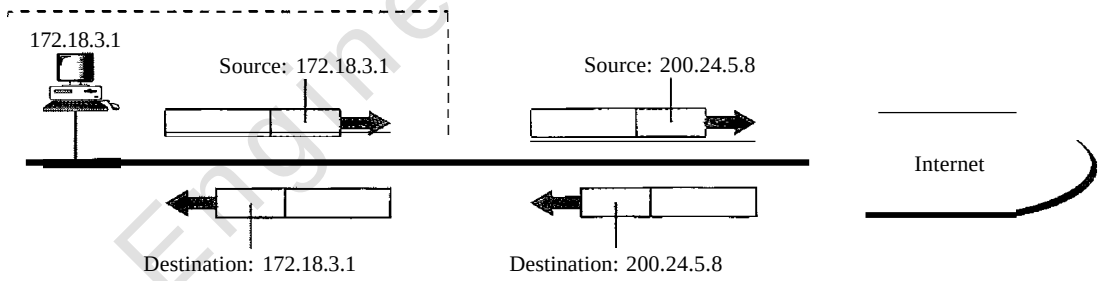
Figure 19.10 *A NAT implementation*



Address Translation

All the outgoing packets go through the NAT router, which replaces the *source address* in the packet with the global NAT address. All incoming packets also pass through the NAT router, which replaces the *destination address* in the packet (the NAT router global address) with the appropriate private address. Figure 19.11 shows an example of address translation.

Figure 19.11 *Addresses in a NAT*

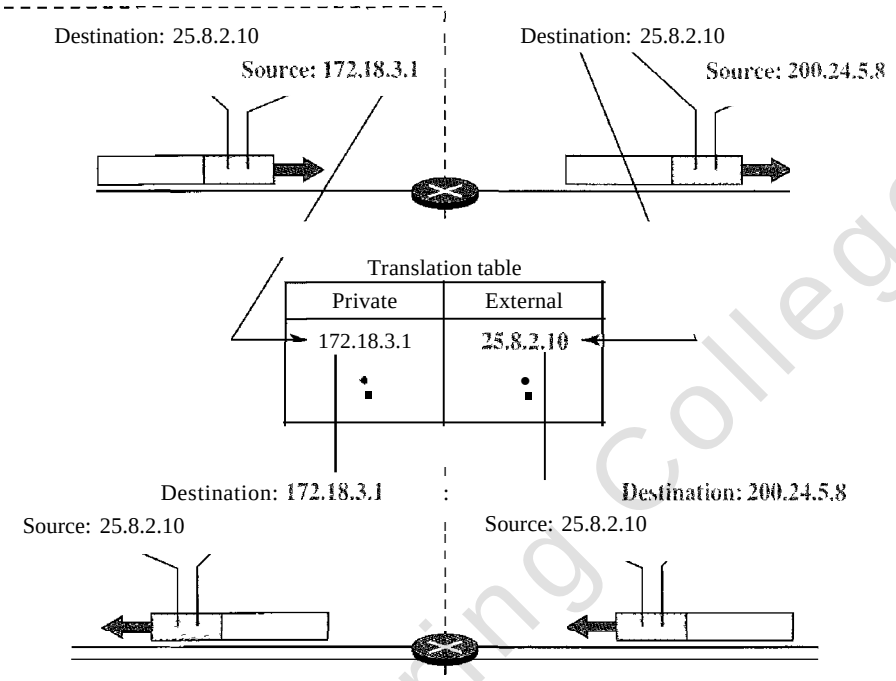


Translation Table

The reader may have noticed that translating the source addresses for outgoing packets is straightforward. But how does the NAT router know the destination address for a packet coming from the Internet? There may be tens or hundreds of private IP addresses, each belonging to one specific host. The problem is solved if the NAT router has a translation table.

Using One IP Address In its simplest fonn, a translation table has only two columns: the private' address and the external address (destination address of the packet). When the router translates the source address of the outgoing packet, it also makes note of the destination address-where the packet is going. When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet. Figure 19.12 shows the idea. Note that the addresses that are changed (translated) are shown in color.

Figure 19.12 *NAT address translation*



In this strategy, communication must always be initiated by the private network. The NAT mechanism described requires that the private network start the communication. As we will see, NAT is used mostly by ISPs which assign one single address to a customer. The customer, however, may be a member of a private network that has many private addresses. In this case, communication with the Internet is always initiated from the customer site, using a client program such as HTTP, TELNET, or FTP to access the corresponding server program. For example, when e-mail that originates from a non-customer site is received by the ISP e-mail server, the e-mail is stored in the mailbox of the customer until retrieved. A private network cannot run a server program for clients outside of its network if it is using NAT technology.

Using a Pool of **IP** Addresses Since the NAT router has only one global address, only one private network host can access the same external host. To remove this restriction, the NAT router uses a pool of global addresses. For example, instead of using only one global address (200.24.5.8), the NAT router can use four addresses (200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11). In this case, four private network hosts can communicate with the same external host at the same time because each pair of addresses defines a connection. However, there are still some drawbacks. In this example, no more than four connections can be made to the same destination. Also, no private-network host can access two external server programs (e.g., HTTP and FfP) at the same time.

Using Both **IP** Addresses and Port Numbers To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table. For example, suppose two hosts with addresses 172.18.3.1 and 172.18.3.2 inside a private network need to access the HTTP server on external host

25.8.3.2. If the translation table has five columns, instead of two, that include the source and destination port numbers of the transport layer protocol, the ambiguity is eliminated. We discuss port numbers in Chapter 23. Table 19.4 shows an example of such a table.

Table 19.4 *Five-column translation table*

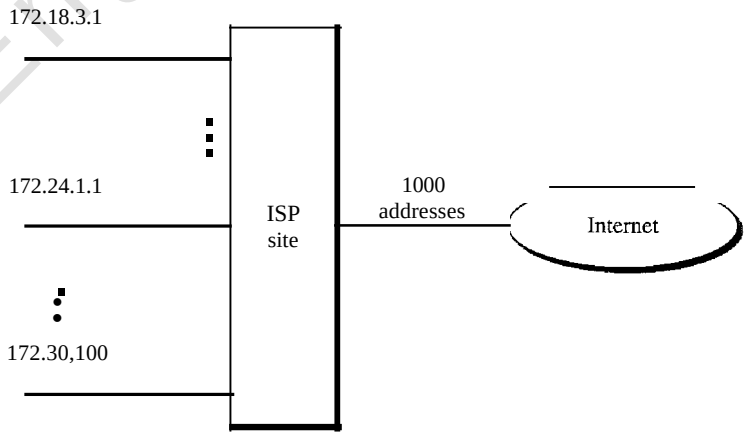
Private Address	*Private Port*	*External Address*	*External Port*	*Transport Protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
...	...	...	...	...

Note that when the response from HTTP comes back, the combination of source address (25.8.3.2) and destination port number (1400) defines the-private network host to which the response should be directed. Note also that for this translation to work, the temporary port numbers (1400 and 1401) must be unique.

NAT and ISP

An ISP that serves dial-up customers can use NAT technology to conserve addresses. For example, suppose an ISP is granted 1000 addresses, but has 100,000 customers. Each of the customers is assigned a private network address. The ISP translates each of the 100,000 source addresses in outgoing packets to one of the 1000 global addresses; it translates the global destination address in incoming packets to the corresponding private address. Figure 19.13 shows this concept.

Figure 19.13 *An ISP and NAT*



19.2 IPv6 ADDRESSES

Despite all short-term solutions, such as classless addressing, Dynamic Host Configuration Protocol (DHCP), discussed in Chapter 21, and NAT, address depletion is still a long-term problem for the Internet. This and other problems in the IP protocol itself,

such as lack of accommodation for real-time audio and video transmission, and encryption and authentication of data for some applications, have been the motivation for IPv6. In this section, we compare the address structure of IPv6 to IPv4. In Chapter 20, we discuss both protocols.

Structure

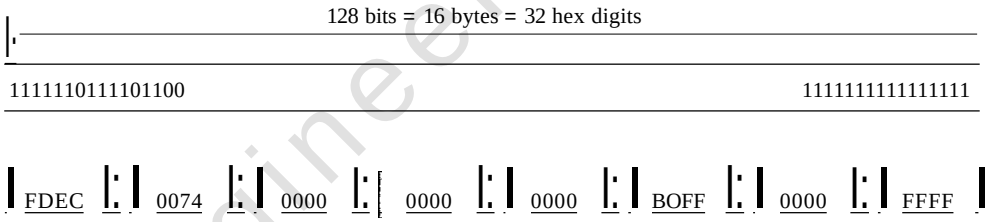
An IPv6 address consists of 16 bytes (octets); it is 128 bits long.

An IPv6 address is 128 bits long.

Hexadecimal Colon Notation

To make addresses more readable, IPv6 specifies hexadecimal colon notation. In this notation, 128 bits is divided into eight sections, each 2 bytes in length. Two bytes in hexadecimal notation requires four hexadecimal digits. Therefore, the address consists of 32 hexadecimal digits, with every four digits separated by a colon, as shown in Figure 19.14.

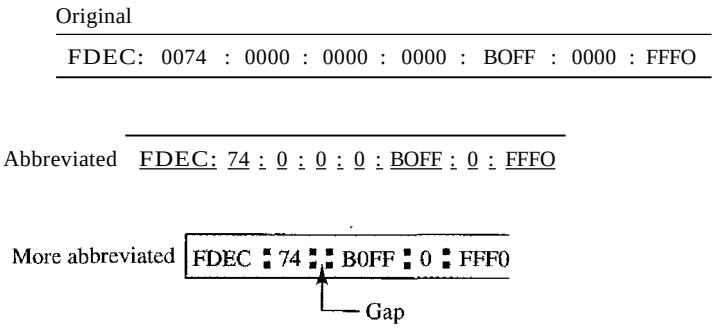
Figure 19.14 *IPv6 address in binary and hexadecimal colon notation*



Abbreviation

Although the IP address, even in hexadecimal format, is very long, many of the digits are zer9s. In this case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted. Only the leading zeros can be dropped, not the trailing zeros (see Figure 19.15).

Figure 19.15 *Abbreviated IPv6 addresses*



Using this form of abbreviation, 0074 can be written as 74, OOOF as F, and 0000 as 0. Note that 3210 cannot be abbreviated. Further abbreviations are possible if there are consecutive sections consisting of zeros only. We can remove the zeros altogether and replace them with a double semicolon. Note that this type of abbreviation is allowed only once per address. **If** there are two runs of zero sections, only one of them can be abbreviated. Reexpansion of the abbreviated address is very simple: Align the unabbreviated portions and insert zeros to get the original expanded address.

Example 19.11

Expand the address 0:15::1:12:1213 to its original.

Solution

We first need to align the left side of the double colon to the left of the original pattern and the right side of the double colon to the right of the original pattern to find now many 0s we need to replace the double colon.

```
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
 0:   15:                   1:   12:1213
```

This means that the original address is

0000:0015:0000:0000:0000:0001:0012:1213

Address Space

IPv6 has a much larger address space; 2^{128} addresses are available. The designers of IPv6 divided the address into several categories. A few leftmost bits, called the *type prefix,* in each address define its category. The type prefix is variable in length, but it is designed such that no code is identical to the first part of any other code. In this way, there is no ambiguity; when an address is given, the type prefix can easily be determined. Table 19.5 shows the prefix for each type of address. The third column shows the fraction of each type of address relative to the whole address space.

Table 19.5 *Type prefixes for 1Pv6 addresses*

Type Prefix	*Type*	*Fraction*
00000000	Reserved	1/256
00000001	Unassigned	1/256
0000001	ISO network addresses	1/128
0000010	IPX (Novell) network addresses	1/128
0000011	Unassigned	1/128
00001	Unassigned	1/32
0001	Reserved	*1/16*
001	Reserved	1/8
010	Provider-based unicast addresses	*1/8*

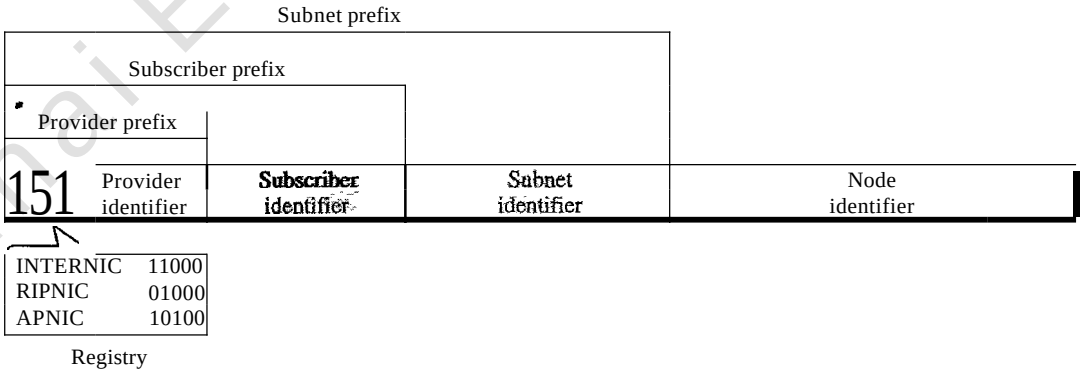
Table 19.5 *Type prefixes for IPv6 addresses (continued)*

Type Prefix	Type	Fraction
011	Unassigned	1/8
100	Geographic-based unicast addresses	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1116
11110	Unassigned	1132
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
11111110 a	Unassigned	1/512
1111 111010	Link local addresses	111024
1111 1110 11	Site local addresses	1/1024
11111111	Multicast addresses	1/256

Unicast Addresses

A **unicast address** defines a single computer. The packet sent to a unicast address must be delivered to that specific computer. IPv6 defines two types of unicast addresses: geographically based and provider-based. We discuss the second type here; the first type is left for future definition. The provider-based address is generally used by a normal host as a unicast address. The address format is shown in Figure 19.16.

Figure 19.16 *Prefixes for provider-based unicast address*



Fields for the provider-based address are as follows:

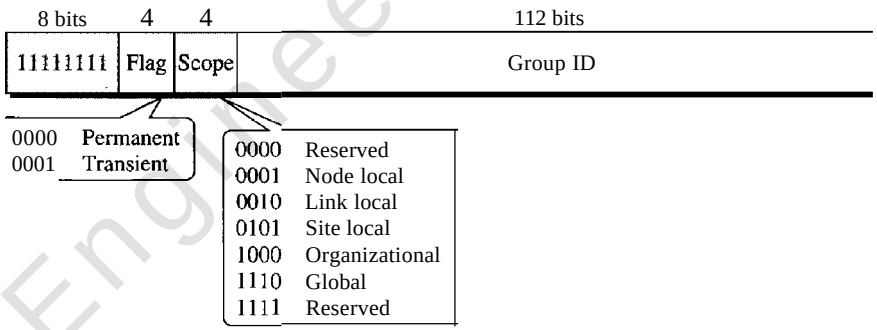
- **Type identifier.** This 3-bit field defines the address as a provider-base.d address.
- **Registry identifier.** This 5-bit field indicates the agency that has registered the address. Currently three registry centers have been defined. INTERNIC (code 11000) is the center for North America; RIPNIC (code 01000) is the center for European registration; and APNIC (code 10100) is for Asian and Pacific countries.

- Provider identifier. This variable-length field identifies the provider for Internet access (such as an ISP). A 16-bit length is recommended for this field.
- Subscriber identifier. When an organization subscribes to the Internet through a provider, it is assigned a subscriber identification. A 24-bit length is recommended for this field.
- Subnet identifier. Each subscriber can have many different subnetworks, and each subnetwork can have an identifier. The subnet identifier defines a specific subnetwork under the territory of the subscriber. A 32-bit length is recommended for this field.
- Node identifier. The last field defines the identity of the node connected to a subnet. A length of 48 bits is recommended for this field to make it compatible with the 48-bit link (physical) address used by Ethernet. In the future, this link address will probably be the same as the node physical address.

Multicast Addresses

Multicast addresses are used to define a group of hosts instead of just one. A packet sent to a multicast address must be delivered to each member of the group. Figure 19.17 shows the format of a multicast address.

Figure 19.17 *Multicast address in IPv6*



The second field is a flag that defines the group address as either permanent or transient. A permanent group address is defined by the Internet authorities and can be accessed at all times. A transient group address, on the other hand, is used only temporarily. Systems engaged in a teleconference, for example, can use a transient group address. The third field defines the scope of the group address. Many different scopes have been defined, as shown in Figure 19.17.

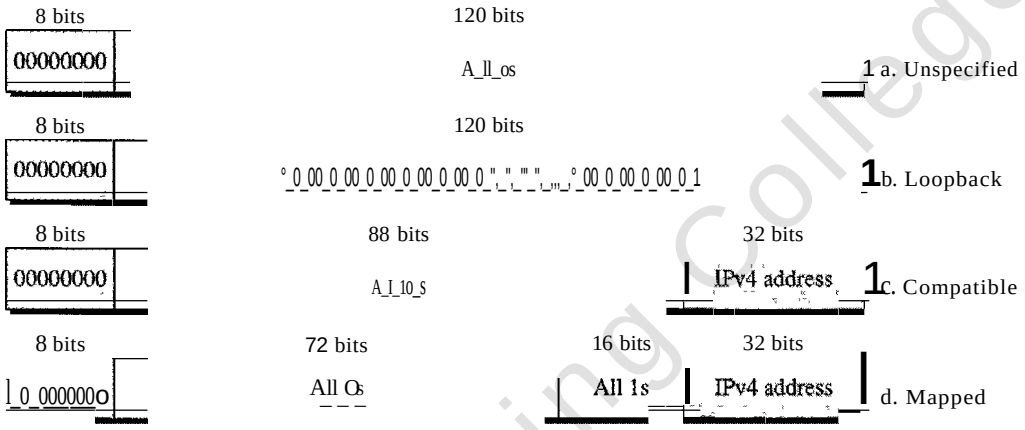
Allycast Addresses

IPv6 also defines anycast addresses. An anycast address, like a multicast address, also defines a group of nodes. However, a packet destined for an anycast address is delivered to only one of the members of the anycast group, the nearest one (the one with the shortest route). Although the definition of an anycast address is still debatable, one possible use is to assign an anycast address to all routers of an ISP that covers a large logical area in the Internet. The routers outside the ISP deliver a packet destined for the ISP to the nearest ISP router. No block is assigned for anycast addresses.

Reserved Addresses

Another category in the address space is the reserved address. These addresses start with eight 0s (type prefix is 00000000). A few subcategories are defined in this category, as shown in Figure 19.18.

Figure 19.18 *Reserved addresses in IPv6*

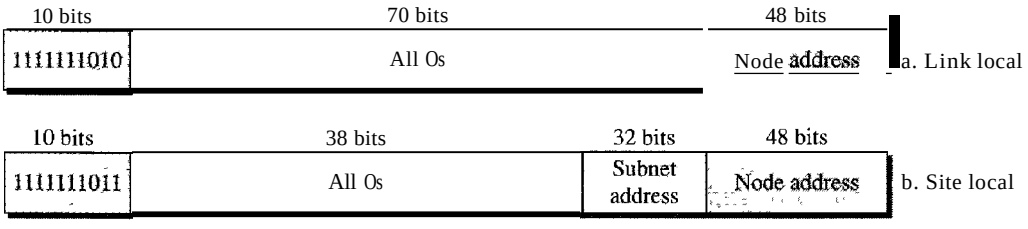


An unspecified address is used when a host does not know its own address and sends an inquiry to find its address. A loopback address is used by a host to test itself without going into the network. A compatible address is used during the transition from IPv4 to IPv6 (see Chapter 20). It is used when a computer using IPv6 wants to send a message to another computer using IPv6, but the message needs to pass through a part of the network that still operates in IPv4. A mapped address is also used during transition. However, it is used when a computer that has migrated to IPv6 wants to send a packet to a computer still using IPv4.

Local Addresses

These addresses are used when an organization wants to use IPv6 protocol without being connected to the global Internet. In other words, they provide addressing for private networks. Nobody outside the organization can send a message to the nodes using these addresses. Two types of addresses are defined for this purpose, as shown in Figure 19.19.

Figure 19.19 *Local addresses in IPv6*



A link local address is used in an isolated subnet; a site local address is used in an isolated site with several subnets.

19.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

IPv4 addresses are discussed in Chapters 4 and 5 of [For06], Chapter 3 of [Ste94], Section 4.1 of [PD03], Chapter 18 of [Sta04], and Section 5.6 of [Tan03]. IPv6 addresses are discussed in Section 27.1 of [For06] and Chapter 8 of [Los04]. A good discussion of NAT can be found in [DutOl].

Sites

- www.ietf.org/rfc.html Information about RFCs

RFCs

A discussion of IPv4 addresses can be found in most of the RFCs related to the IPv4 protocol:

760,781,791,815,1025,1063,1071,1141,1190, 1191, 1624,2113

A discussion of 1Pv6 addresses can be found in most of the RFCs related to IPv6 protocol:

1365,1550,1678,1680,1682,1683,1686,1688,1726, 1752, 1826, 1883, 1884,1886,1887, 1955,2080,2373,2452,2463,2465,2466,2472,2492,2545,2590

A discussion of NAT can be found in

1361,2663,2694

19.4 KEY TERMS

address aggregation
address space
anycast address
binary notation
class A address
class B address
class C address
class D address
class E address
classful addressing
classless addressing
classless interdomain routing (CIDR)

compatible address
dotted-decimal notation
default mask
hexadecimal colon notation
hostid
IP address
IPv4 address
IPv6 address
link local address
mapped address
mask
multicast address
netid
network address
network address translation (NAT)
prefix
reserved address
site local address
subnet
subnet mask
subnetting
suffix
supernet
supernet mask
supernetting
unicast address
unspecified address

19.5 SUMMARY

- At the network layer, a global identification system that uniquely identifies every host and router is necessary for delivery of a packet from host to host.
- An IPv4 address is 32 bits long and uniquely and universally defines a host or router on the Internet.
- In classful addressing, the portion of the IP address that identifies the network is called the netid.
- In classful addressing, the portion of the IP address that identifies the host or router on the network is called the hostid.
- An IP address defines a device's connection to a network.
- There are five classes in IPv4 addresses. Classes A, B, and C differ in the number of hosts allowed per network. Class D is for multicasting and Class E is reserved.
- The class of an address is easily determined by examination of the first byte.
- Addresses in classes A, B, or C are mostly used for unicast communication.
- Addresses in class D are used for multicast communication.
- Subnetting divides one large network into several smaller ones, adding an intermediate level of hierarchy in IP addressing.
- Supernetting combines several networks into one large one.
- In classless addressing, we can divide the address space into variable-length blocks.
- There are three restrictions in classless addressing:
  a. The number of addresses needs to be a power of 2.
  b. The mask needs to be included in the address to define the block.
  c. The starting address must be divisible by the number of addresses in the block.
- The mask in classless addressing is expressed as the prefix length *(/n)* in CIDR notation.

- To find the first address in a block, we set the rightmost 32 - n bits to 0.
- To find the number of addresses in the block, we calculate 2^{32-n}, where n is the prefix length.
- To find the last address in the block, we set the rightmost 32 - n bits to 1.
- Subnetting increases the value of n.
- The global authority for address allocation is ICANN. ICANN normally grants large blocks of addresses to ISPs, which in tum grant small subblocks to individual customers.
- IPv6 addresses use hexadecimal colon notation with abbreviation methods available.
- There are three types of addresses in IPv6: unicast, anycast, and multicast.
- In an IPv6 address, the variable type prefix field defines the address type or purpose.

19.6 PRACTICE SET

Review Questions

1. What is the number of bits in an IPv4 address? What is the number of bits in an IPv6 address?
2. What is dotted decimal notation in IPv4 addressing? What is the number of bytes in an IPv4 address represented in dotted decimal notation? What is hexadecimal notation in IPv6 addressing? What is the number of digits in an IPv6 address represented in hexadecimal notation?
3. What are the differences between classful addressing and classless addressing in IPv4?
4. List the classes in classful addressing and define the application of each class (unicast, multicast, broadcast, or reserve).
5. Explain why most of the addresses in class A are wasted. Explain why a medium-size or large-size corporation does not want a block of class C addresses.
6. What is a mask in IPv4 addressing? What is a default mask in IPv4 addressing?
7. What is the network address in a block of addresses? How can we find the network address if one of the addresses in a block is given?
8. Briefly define subnetting and supemetting. How do the subnet mask and supemet mask differ from a default mask in classful addressing?
9. How can we distinguish a multicast address in IPv4 addressing? How can we do so in IPv6 addressing?
10. What is NAT? How can NAT help in address depletion?

Exercises

II. What is the address space in each of the following systems?
   a. A system with 8-bit addresses
   b. A system with 16-bit addresses
   c. A system with 64-bit addresses

12. An address space has a total of 1024 addresses. How many bits are needed to represent an address?
13. An address space uses the three symbols 0, 1, and 2 to represent addresses. If each address is made of 10 symbols, how many addresses are available in this system?
14. Change the following IP addresses from dotted-decimal notation to binary notation.
    a. 114.34.2.8
    b. 129.14.6.8
    c. 208.34.54.12
    d. 238.34.2.1
15. Change the following IP addresses from binary notation to dotted-decimal notation.
    a. 01111111 11110000 01100111 01111101
    b. 10101111 11000000 11111000 00011101
    c. 11011111 10110000 00011111 01011101
    d. 11101111 11110111 11000111 00011101
16. Find the class of the following IP addresses.
    a. 208.34.54.12
    b. 238.34.2.1
    c. 114.34.2.8
    d. 129.14.6.8
17. Find the class of the following IP addresses.
    a. 11110111 11110011 10000111 11011101
    b. 10101111 11000000 11110000 00011101
    c. 11011111 10110000 00011111 01011101
    d. 11101111 11110111 11000111 00011101
18. Find the netid and the hostid of the following IP addresses.
    a. 114.34.2.8
    b. 132.56.8.6
    c. 208.34.54.12
19. In a block of addresses, we know the IP address of one host is 25.34.12.56/16. What are the first address (network address) and the last address (limited broadcast address) in this block?
20. In a block of addresses, we know the IP address of one host is 182.44.82.16/26. What are the first address (network address) and the last address in this block?
21. An organization is granted the block 16.0.0.0/8. The administrator wants to create 500 fixed-length subnets.
    a. Find the subnet mask.
    b. Find the number of addresses in each subnet.
    c. Find the first and last addresses in subnet 1.
    d. Find the first and last addresses in subnet 500.

22. An organization is granted the block 130.56.0.0/16. The administrator wants to create 1024 subnets.
    a. Find the subnet mask.
    b. Find the number of addresses in each subnet.
    c. Find the first and last addresses in subnet 1.
    d. Find the first and last addresses in subnet 1024.
23. An organization is granted the block 211.17.180.0/24. The administrator wants to create 32 subnets.
    a. Find the subnet mask.
    b. Find the number of addresses in each subnet.
    c. Find the first and last addresses in subnet 1.
    d. Find the first and last addresses in subnet 32.
24. Write the following masks in slash notation *(In).*
    a. 255.255.255.0
    b. 255.0.0.0
    c. 255.255.224.0
    d. 255.255.240.0
25. Find the range of addresses in the following blocks.
    a. 123.56.77.32/29
    b. 200.17.21.128/27
    c. 17.34.16.0/23
    d. 180.34.64.64/30
26. An ISP is granted a block of addresses starting with 150.80.0.0/16. The ISP wants to distribute these blocks to 2600 customers as follows.
    a. The first group has 200 medium-size businesses; each needs 128 addresses.
    b. The second group has 400 small businesses; each needs 16 addresses.
    c. The third group has 2000 households; each needs 4 addresses.

    Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.
27. An ISP is granted a block of addresses starting with 120.60.4.0/22. The ISP wants to distribute these blocks to 100 organizations with each organization receiving just eight addresses. Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.
28. An ISP has a block of 1024 addresses. It needs to divide the addresses among 1024 customers. Does it need subnetting? Explain your answer.
29. Show the shortest form of the following addresses.
    a. 2340: lABC: 119A:AOOO:0000:0000:0000:0000
    b. 0000:00AA:0000:0000:0000:0000: 119A:A231
    c. 2340:0000:0000:0000:0000: 119A:AOO1:0000
    d. 0000:0000:0000:2340:0000:0000:0000:0000

30. Show the original (unabbreviated) form of the following addresses.
    a. 0::0
    b.O:AA::O
    c. 0: 1234::3
    d. 123::1:2
31. What is the type of each of the following addresses?
    a. FE80::12
    b. FECO: :24A2
    c. FF02::0
    d. 0::01
32. What is the type of each of the following addresses?
    a. 0::0
    b. 0: :FFFF:O:O
    c. 582F:1234::2222
    d. 4821::14:22
    e. 54EF::A234:2
33. Show the provider prefix (in hexadecimal colon notation) of an address assigned to a subscriber if it is registered in the United States with ABC1 as the provider identification.
34. Show in hexadecimal colon notation the IPv6 address
    a. Compatible to the IPv4 address 129.6.12.34
    b. Mapped to the IPv4 address 129.6.12.34
35. Show in hexadecimal colon notation
    a. The link local address in which the node identifier is 0:: 123/48
    b. The site local address in which the node identifier is 0:: 123/48
36. Show in hexadecimal colon notation the permanent multicast address used in a link local scope.
37. A host has the address 581E: 1456:2314:ABCD:: 1211. If the node identification is 48 bits, find the address of the subnet to which the host is attached.
38. A site with 200 subnets has the class B address of 132.45.0.0. The site recently migrated to IPv6 with the subscriber prefix 581E:1456:2314::ABCD/80. Design the subnets and define the subnet addresses, using a subnet identifier of 32 bits.

Research Activities

39. Find the block of addresses assigned to your organization or institution.
40. If you are using an ISP to connect from your home to the Internet, find the name of the ISP and the block of addresses assigned to it.
41. Some people argue that we can consider the whole address space as one single block in which each range of addresses is a subblock to this single block. Elaborate on this idea. What happens to subnetting if we accept this concept?
42. Is your school or organization using a classful address? If so, find out the class of the address.

CHAPTER 20

Network Layer: Internet Protocol

In the Internet model, the main network protocol is the **Internet Protocol** (IP). In this chapter, we first discuss internetworking and issues related to the network layer protocol in general.

We then discuss the current version of the Internet Protocol, version 4, or IPv4. This leads us to the next generation of this protocol, or IPv6, which may become the dominant protocol in the near future.

Finally, we discuss the transition strategies from IPv4 to IPv6. Some readers may note the absence of IPv5. IPv5 is an experimental protocol, based mostly on the OSI model that never materialized.

20.1 INTERNETWORKING

The physical and data link layers of a network operate locally. These two layers are jointly responsible for data delivery on the network from one node to the next, as shown in Figure 20.1.

This internetwork is made of five networks: four LANs and one WAN. If host A needs to send a data packet to host D, the packet needs to go first from A to R1 (a switch or router), then from R1 to R3, and finally from R3 to host D. We say that the data packet passes through three links. In each link, two physical and two data link layers are involved.

However, there is a big problem here. When data arrive at interface f1 of R1, how does RI know that interface f3 is the outgoing interface? There is no provision in the data link (or physical) layer to help R1 make the right decision. The frame does not carry any routing information either. The frame contains the MAC address of A as the source and the MAC address of R1 as the destination. For a LAN or a WAN, delivery means carrying the frame through one link, and not beyond.

Need for Network Layer

To solve the problem of delivery through several links, the network layer (or the internetwork layer, as it is sometimes called) was designed. The network layer is responsible for host-to-host delivery and for routing the packets through the routers or switches. Figure 20.2 shows the same internetwork with a network layer added.

Figure 20.1 *Links between two hosts*

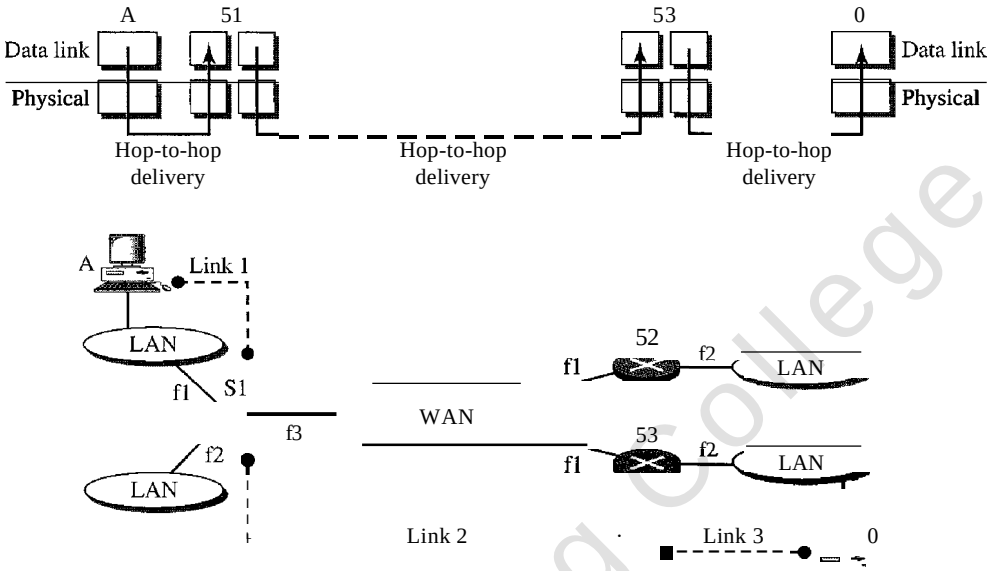


Figure 20.2 *Network layer in an internetwork*

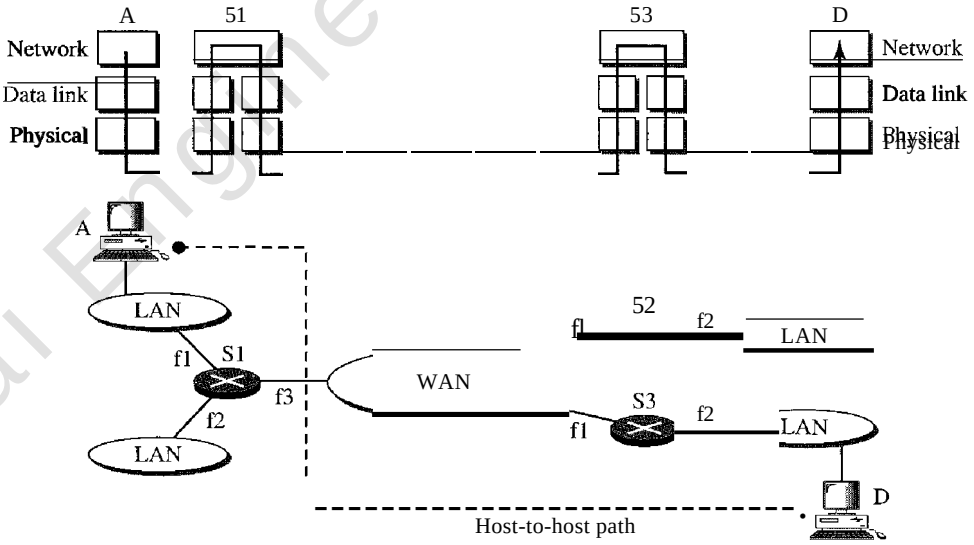
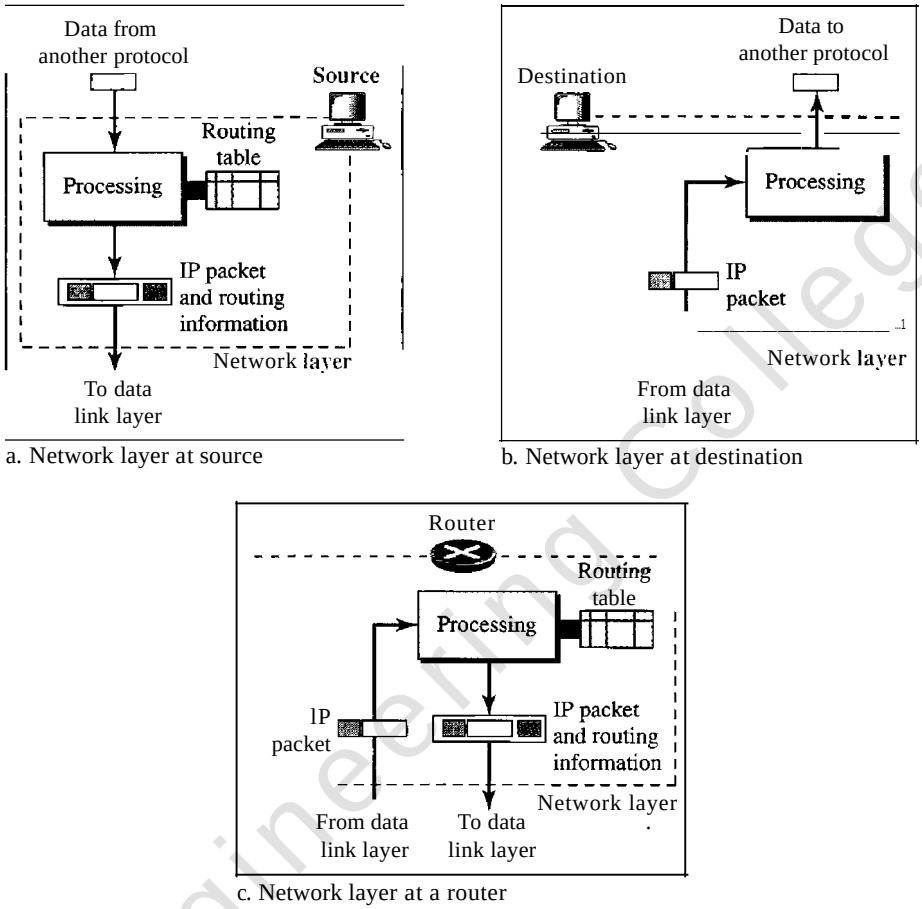


Figure 20.3 shows the general idea of the functionality of the network layer at a source, at a router, and at the destination. The network layer at the source is responsible for creating a packet from the data coming from another protocol (such as a transport layer protocol or a routing protocol). The header of the packet contains, among other information, the logical addresses of the source and destination. The network layer is responsible for checking its routing table to find the routing information (such as the outgoing interface of the packet or the physical address of the next node). If the packet is too large, the packet is fragmented (fragmentation is discussed later in this chapter).

Figure 20.3 *Network layer at the source, router, and destination*



a. Network layer at source

b. Network layer at destination

c. Network layer at a router

The network layer at the switch or router is responsible for routing the packet. When a packet arrives, the router or switch consults its routing table and finds the interface from which the packet must be sent. The packet, after some changes in the header, with the routing infonnation is passed to the data link layer again.

The network layer at the destination is responsible for address verification; it makes sure that the destination address on the packet is the same as the address of the host. If the packet is a fragment, the network layer waits until all fragments have arrived, and then reassembles them and delivers the reassembled packet to the transport layer.

Internet as a Datagram Network

The Internet, at the network layer, is a packet-switched network. We discussed switching in Chapter 8. We said that, in general, switching can be divided into three broad categories: circuit switching, packet switching, and message switching. Packet switching uses either the virtual circuit approach or the datagram approach.

The Internet has chosen the datagram approach to switching in the network layer. It uses the universal addresses defined in the network layer to route packets from the source to the destination.

Switching at the network layer in the Internet uses the datagram approach to packet switching.

Internet as a Connectionless Network

Delivery of a packet can be accomplished by using either a connection-oriented or a connectionless network service. In a connection-oriented service, the source first makes a connection with the destination before sending a packet. When the connection is established, a sequence of packets from the same source to the same destination can be sent one after another. In this case, there is a relationship between packets. They are sent on the same path in sequential order. A packet is logically connected to the packet traveling before it and to the packet traveling after it. When all packets of a message have been delivered, the connection is terminated.

In a connection-oriented protocol, the decision about the route of a sequence of packets with the same source and destination addresses can be made only once, when the connection is established. Switches do not recalculate the route for each individual packet. This type of service is used in a virtual-circuit approach to packet switching such as in Frame Relay and ATM.

In conneetionless service, the network layer protocol treats each packet independently, with each packet having no relationship to any other packet. The packets in a message mayor may not travel the same path to their destination. This type of service is used in the datagram approach to packet switching. The Internet has chosen this type of service at the network layer.

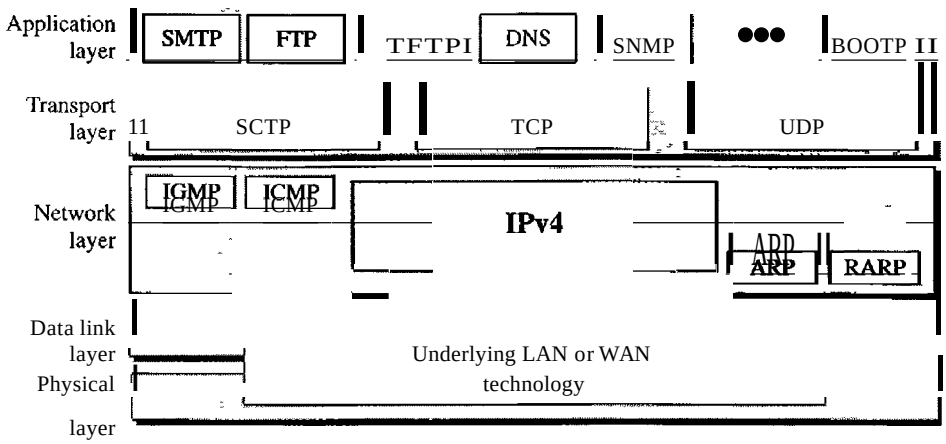
The reason for this decision is that the Internet is made of so many heterogeneous networks that it is almost impossible to create a connection from the source to the destination without knowing the nature of the networks in advance.

Communication at the network layer in the Internet is connectionless.

20.2 IPv4

The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/IP protocols. Figure 20.4 shows the position of IPv4 in the suite.

Figure 20.4 *Position ofIPv4 in TCPIIP protocol suite*



IPv4 is an unreliable and connectionless datagram protocol-a best-effort delivery service. The term *best-effort* means that IPv4 provides no error control or flow control (except for error detection on the header). IPv4 assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.

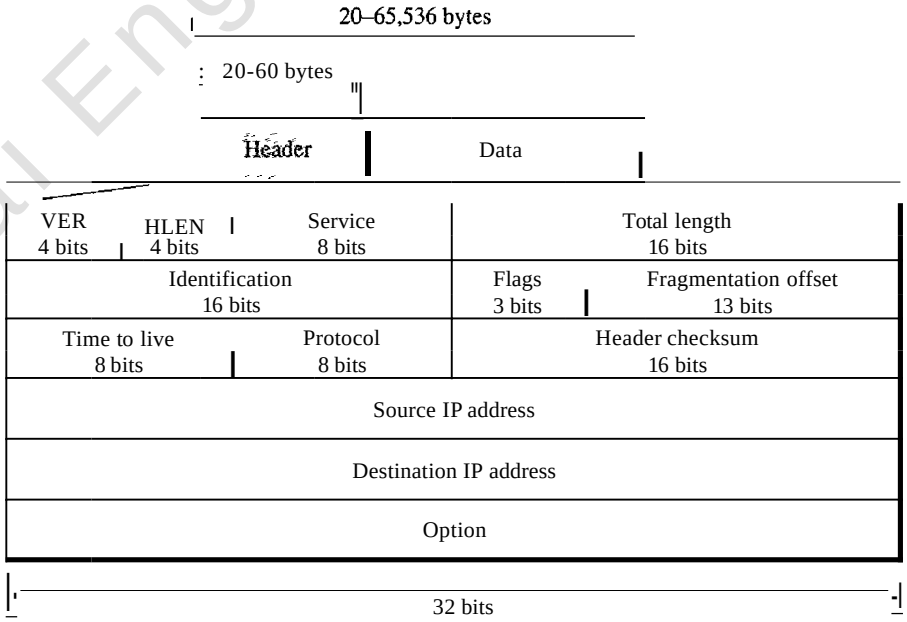
If reliability is important, IPv4 must be paired with a reliable protocol such as TCP. An example of a more commonly understood best-effort delivery service is the post office. The post office does its best to deliver the mail but does not always succeed. If an unregistered letter is lost, it is up to the sender or would-be recipient to discover the loss and rectify the problem. The post office itself does not keep track of every letter and cannot notify a sender of loss or damage.

IPv4 is also a connectionless protocol for a packet-switching network that uses the datagram approach (see Chapter 8). This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Also, some could be lost or corrupted during transmission. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

Datagram

Packets in the IPv4 layer are called datagrams. Figure 20.5 shows the IPv4 datagram format.

Figure 20.5 *IPv4 datagramfannat*

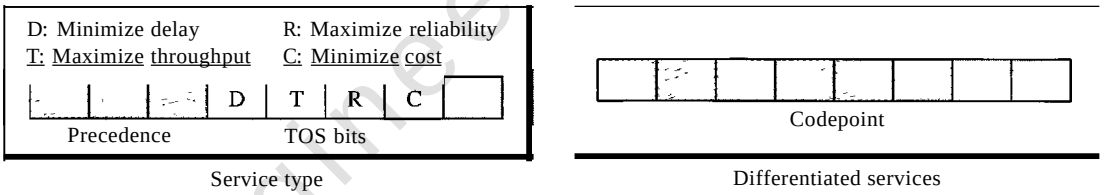


A datagram is a variable-length packet consisting of two parts: header and data. The header is 20 to 60 bytes in length and contains information essential to routing and

delivery. It is customary in *TCP/IP* to show the header in 4-byte sections. A brief description of each field is in order.

- Version (VER). This 4-bit field defines the version of the IPv4 protocol. Currently the version is 4. However, version 6 (or IPng) may totally replace version 4 in the future. This field tells the IPv4 software running in the processing machine that the datagram has the format of version 4. All fields must be interpreted as specified in the fourth version of the protocol. If the machine is using some other version of IPv4, the datagram is discarded rather than interpreted incorrectly.
- Header length (HLEN). This 4-bit field defines the total length of the datagram header in 4-byte words. This field is needed because the length of the header is variable (between 20 and 60 bytes). When there are no options, the header length is 20 bytes, and the value of this field is 5 (5 x 4 = 20). When the option field is at its maximum size, the value of this field is 15 (15 x 4 = 60).
- Services. IETF has changed the interpretation and name of this 8-bit field. This field, previously called service type, is now called differentiated services. We show both interpretations in Figure 20.6.

Figure 20.6 *Service type or differentiated services*



1. Service Type
   In this interpretation, the first 3 bits are called precedence bits. The next 4 bits are called type of service (TOS) bits, and the last bit is not used.
   a. Precedence is a 3-bit subfield ranging from 0 (000 in binary) to 7 (111 in binary). The precedence defines the priority of the datagram in issues such as congestion. If a router is congested and needs to discard some datagrams, those datagrams with lowest precedence are discarded first. Some datagrams in the Internet are more important than others. For example, a datagram used for network management is much more urgent and important than a datagram containing optional information for a group.

The precedence subfield was part of version 4, but never used.

   b. TOS bits is a 4-bit subfield with each bit having a special meaning. Although a bit can be either 0 or 1, one and only one of the bits can have the value of 1 in each datagram. The bit patterns and their interpretations are given in Table 20.1. With only 1 bit set at a time, we can have five different types of services.

Table 20.1 *Types of service*

TOS Bits	*Description*
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

Application programs can request a specific type of service. The defaults for some applications are shown in Table 20.2.

Table 20.2 *Default types of service*

Protocol	*TOS Bits*	*Description*
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

It is clear from Table 20.2 that interactive activities, activities requiring immediate attention, and activities requiring immediate response need minimum delay. Those activities that send bulk data require maximum throughput. Management activities need maximum reliability. Background activities need minimum cost.

2. Differentiated Services

In this interpretation, the first 6 bits make up the codepoint subfield, and the last 2 bits are not used. The codepoint subfield can be used in two different ways.

a. When the 3 rightmost bits are 0s, the 3 leftmost bits are interpreted the same as the precedence bits in the service type interpretation. In other words, it is compatible with the old interpretation.

b. When the 3 rightmost bits are not all Os, the 6 bits define 64 services based on the priority assignment by the Internet or local authorities according to Table 20.3. The first category contains 32 service types; the second and the third each contain 16. The first category (numbers 0, 2,4, ... ,62) is assigned by the Internet authorities (IETF). The second category (3, 7, 11, 15, , 63) can be used by local authorities (organizations). The third category (1, 5, 9, ,61) is temporary and can be used for experimental purposes. Note that the numbers are not contiguous. If they were, the first category would range from 0 to 31, the second from 32 to 47, and the third from 48 to 63. This would be incompatible with the TOS interpretation because XXXOOO (which includes 0, 8, 16, 24, 32, 40, 48, and 56) would fall into all three categories. Instead, in this assignment method all these services belong to category 1. Note that these assignments have not yet been finalized.

Table 20.3 *Values for codepoints*

Category	*Codepoint*	*Assigning Authority*
1	XXXXXO	Internet
2	XXXXll	Local
3	XXXXOI	Temporary or experimental

- Total length. This is a In-bit field that defines the total length (header plus data) of the IPv4 datagram in bytes. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by 4.

$$\text{Length of data} = \text{total length} - \text{header length}$$

Since the field length is 16 bits, the total length of the IPv4 datagram is limited to 65,535 ($2^{16} - 1$) bytes, of which 20 to 60 bytes are the header and the rest is data from the upper layer.

The total length field defines the total length of the datagram including the header.

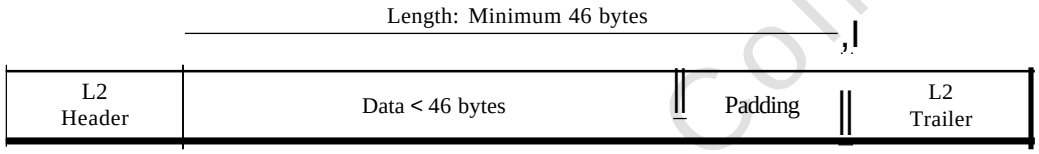
Though a size of 65,535 bytes might seem large, the size of the IPv4 datagram may increase in the near future as the underlying technologies allow even more throughput (greater bandwidth).

When we discuss fragmentation in the next section, we will see that some physical networks are not able to encapsulate a datagram of 65,535 bytes in their frames. The datagram must be fragmented to be able to pass through those networks.

One may ask why we need this field anyway. When a machine (router or host) receives a frame, it drops the header and the trailer, leaving the datagram. Why include an extra field that is not needed? The answer is that in many cases we really do not need the value in this field. However, there are occasions in which the

datagram is not the only thing encapsulated in a frame; it may be that padding has been added. For example, the Ethernet protocol has a minimum and maximum restriction on the size of data that can be encapsulated in a frame (46 to 1500 bytes). If the size of an IPv4 datagram is less than 46 bytes, some padding will be added to meet this requirement. In this case, when a machine decapsulates the datagram, it needs to check the total length field to determine how much is really data and how much is padding (see Figure 20.7).

Figure 20.7 *Encapsulation of a small datagram in an Ethernet frame*

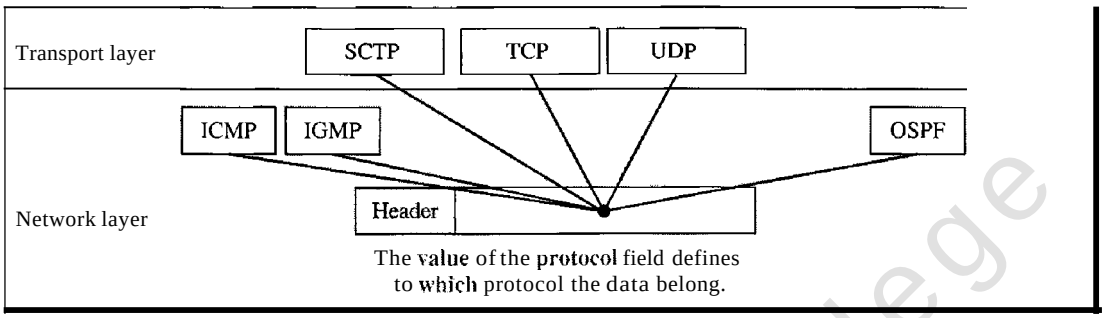


- Identification. This field is used in fragmentation (discussed in the next section).
- Flags. This field is used in fragmentation (discussed in the next section).
- Fragmentation offset. This field is used in fragmentation (discussed in the next section).
- Time to live. A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero. However, for this scheme, all the machines must have synchronized clocks and must know how long it takes for a datagram to go from one machine to another. Today, this field is used mostly to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately 2 times the maximum number of routes between any two hosts. Each router that processes the datagram decrements this number by 1. If this value, after being decremented, is zero, the router discards the datagram.

  This field is needed because routing tables in the Internet can become corrupted. A datagram may travel between two or more routers for a long time without ever getting delivered to the destination host. This field limits the lifetime of a datagram.

  Another use of this field is to intentionally limit the journey of the packet. For example, if the source wants to confine the packet to the local network, it can store 1 in this field. When the packet arrives at the first router, this value is decremented to 0, and the datagram is discarded.
- Protocol. This 8-bit field defines the higher-level protocol that uses the services of the IPv4 layer. An IPv4 datagram can encapsulate data from several higher-level protocols such as TCP, UDP, ICMP, and IGMP. This field specifies the final destination protocol to which the IPv4 datagram is delivered. In other words, since the IPv4 protocol carries data from different other protocols, the value of this field helps the receiving network layer know to which protocol the data belong (see Figure 20.8).

Figure 20.8 *Protocol field and encapsulated data*



The value of this field for each higher-level protocol is shown in Table 20.4.

Table 20.4 *Protocol values*

Value	*Protocol*
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

- Checksum. The checksum concept and its calculation are discussed later in this chapter.
- Source address. This 32-bit field defines the IPv4 address of the source. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.
- Destination address. This 32-bit field defines the IPv4 address of the destination. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

Example 20.1

An IPv4 packet has arrived with the first 8 bits as shown:

01000010

The receiver discards the packet. Why?

Solution

There is an elTOr in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length (2 X 4 = 8). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example 20.2

In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

Solution

The HLEN value is 8, which means the total number of bytes in the header is 8 x 4, or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

Example 20.3

In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is Ox0028. How many bytes of data are being carried by this packet?

Solution

The HLEN value is 5, which means the total number of bytes in the header is 5 x 4, or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40 - 20).

Example 20.4

An IPv4 packet has arrived with the first few hexadecimal digits as shown.

Ox45000028000100000102 ...

How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

Solution

To find the time-to-live field, we skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper-layer protocol is IGMP (see Table 20.4).

Fragmentation

A datagram can travel through different networks. Each router decapsulates the IPv4 datagram from the frame it receives, processes it, and then encapsulates it in another frame. The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled. The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel. For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU)

Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field. In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network (see Figure 20.9).

The value of the MTU depends on the physical network protocol. Table 20.5 shows the values for some protocols.

Figure 20.9 *Maximum transfer unit (MTU)*

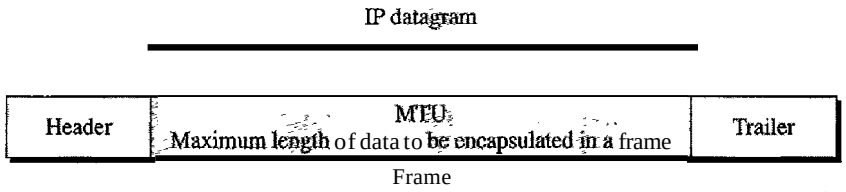


Table 20.5 *MTUs for some networks*

Protocol	*MTU*
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

To make the IPv4 protocol independent of the physical network, the designers decided to make the maximum length of the IPv4 datagram equal to 65,535 bytes. This makes transmission more efficient if we use a protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible to pass through these networks. This is called fragmentation.

The source usually does not fragment the IPv4 packet. The transport layer will instead segment the data into a size that can be accommodated by IPv4 and the data link layer in use.

When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but with some changed. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram can be fragmented several times before it reaches the final destination.

In IPv4, a datagram can be fragmented by the source host or any router in the path although there is a tendency to limit fragmentation only at the source. The reassembly of the datagram, however, is done only by the destination host because each fragment becomes an independent datagram. Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the reassembly at the final destination. An even stronger objection to reassembling packets during the transmission is the loss of efficiency it incurs.

When a datagram is fragmented, required parts of the header must be copied by all fragments. The option field mayor may not be copied, as we will see in the next section. The host or router that fragments a datagram must change the values of three fields:

flags, fragmentation offset, and total length. The rest of the fields must be copied. Of course, the value of the checksum must be recalculated regardless of fragmentation.

Fields Related to Fragmentation

The fields that are related to fragmentation and reassembly of an IPv4 datagram are the identification, flags, and fragmentation offset fields.

- Identification. This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IPv4 protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.
- Flags. This is a 3-bit field. The first bit is reserved. The second bit is called the *do not fragment* bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (see Chapter 21). If its value is 0, the datagram can be fragmented if necessary. The third bit is called the *more fragment* bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment (see Figure 20.10).

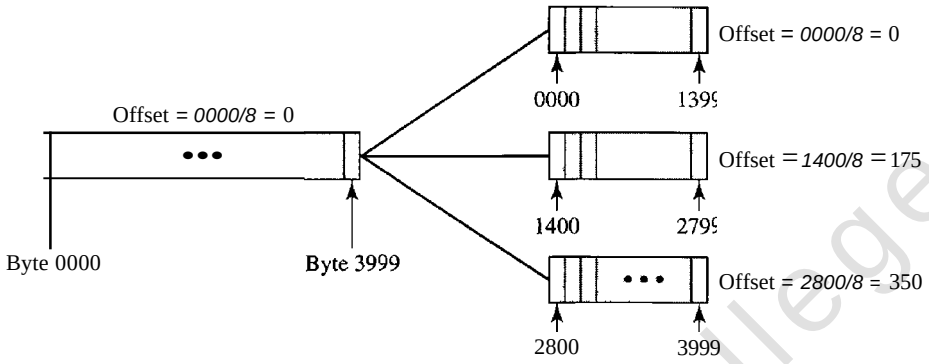
Figure 20.10 *Flags used in fragmentation*

- Fragmentation offset. This 13-bit field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes. Figure 20.11 shows a datagram with a data size of 4000 bytes fragmented into three fragments.

  The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 = 0. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 = 350.

  Remember that the value of the offset is measured in units of 8 bytes. This is done because the length of the offset field is only 13 bits and cannot represent a

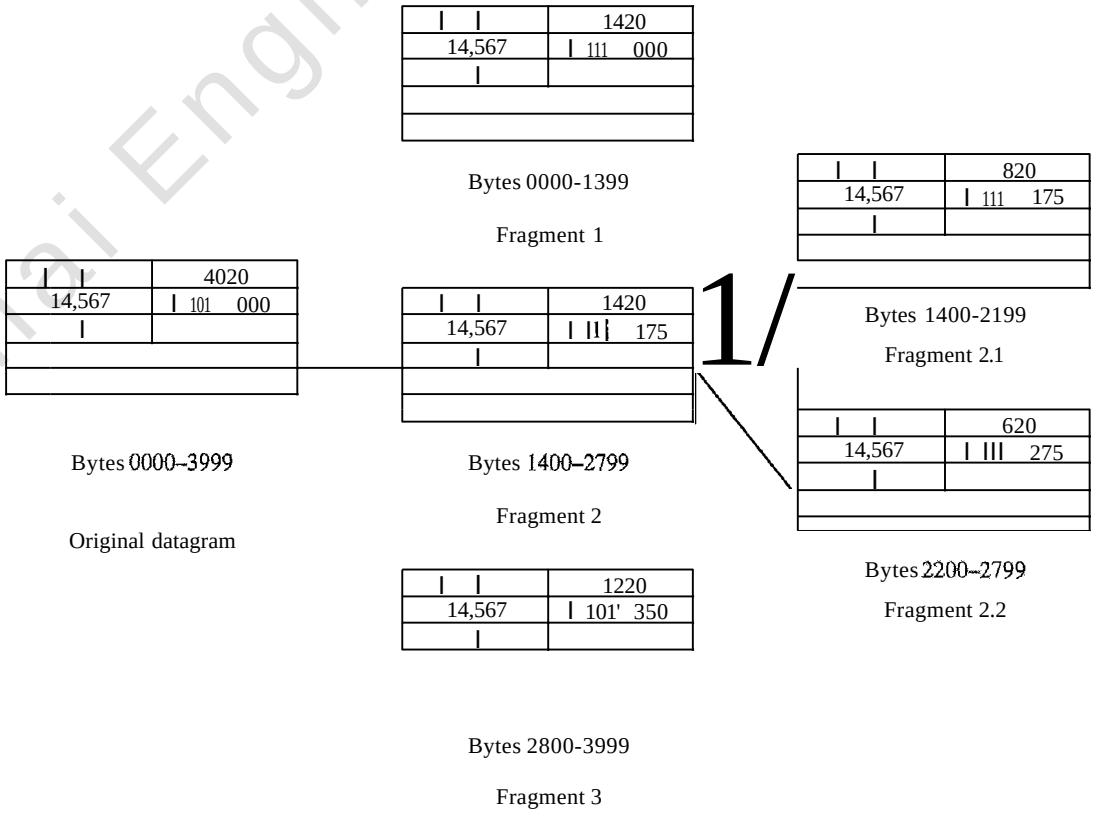
Figure 20.11 *Fragmentation example*



sequence of bytes greater than 8191. This forces hosts or routers that fragment datagrams to choose a fragment size so that the first byte number is divisible by 8.

Figure 20.12 shows an expanded view of the fragments in Figure 20.11. Notice the value of the identification field is the same in all fragments. Notice the value of the flags field with the *more* bit set for all fragments except the last. Also, the value of the offset field for each fragment is shown.

Figure 20.12 *Detailedfragmentation example*



The figure also shows what happens if a fragment itself is fragmented. In this case the value of the offset field is always relative to the original datagram. For example, in the figure, the second fragment is itself fragmented later to two fragments of 800 bytes and 600 bytes, but the offset shows the relative position of the fragments to the original data.

It is obvious that even if each fragment follows a different path and arrives out of order, the final destination host can reassemble the original datagram from the fragments received (if none of them is lost) by using the following strategy:

1. The first fragment has an offset field value of zero.
2. Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.
3. Divide the total length of the first and second fragments by 8. The third fragment has an offset value equal to that result.
4. Continue the process. The last fragment has a *more* bit value of 0.

Example 20.5

A packet has arrived with an *M* bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the *M* bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

Example 20.6

A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See Example 20.7.

Example 20.7

A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, Or a middle fragment?

Solution

Because the *M* bit is l, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 20.8

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

Example 20.9

A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the tota1length field is 100. What are the numbers of the first byte and the last byte?

Solution

The first byte number is 100 x 8 = 800. The total length is 100 bytes, and the header length is 20 bytes (5 x 4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

Checksum

We discussed the general idea behind the checksum and how it is calculated in Chapter 10. The implementation of the checksum in the IPv4 packet follows the same principles. First, the value of the checksum field is set to 0. Then the entire header is divided into 16-bit sections and added together. The result (sum) is complemented and inserted into the checksum field.

The checksum in the IPv4 packet covers only the header, not the data. There are two good reasons for this. First, all higher-level protocols that encapsulate data in the IPv4 datagram have a checksum field that covers the whole packet. Therefore, the checksum for the IPv4 datagram does not have to check the encapsulated data. Second, the header of the IPv4 packet changes with each visited router, but the data do not. So the checksum includes only the part that has changed. If the data were included, each router must recalculate the checksum for the whole packet, which means an increase in processing time.

Example 20.10

Figure 20.13 shows an example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. The result is inserted in the checksum field.

Options

The header of the IPv4 datagram is made of two parts: a fixed palt and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options that can be a maximum of 40 bytes.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IPv4 header, option processing is required of the IPv4 software. This means that all implementations must be able to handle options if they are present in the header.

The detailed discussion of each option is beyond the scope of this book. We give the taxonomy of options in Figure 20.14 and briefly explain the purpose of each.

No Operation

A **no-operation option** is a I-byte option used as a filler between options.

Figure 20.13 *Example of checksum calculation in IPv4*

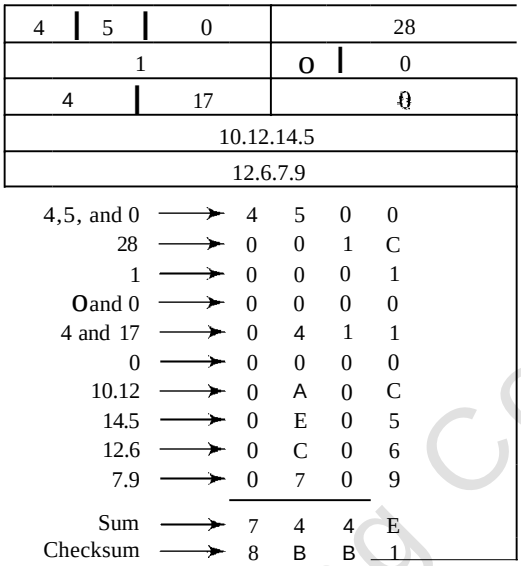
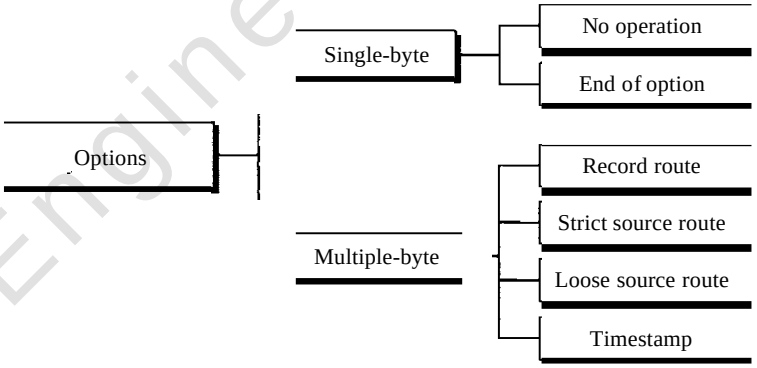


Figure 20.14 *Taxonomy of options in IPv4*



End of Option

An end-of-option option is a I-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

Record Route

A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

Strict Source Route

A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful

for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput. Alternatively, it may choose a route that is safer or more reliable for the sender's purpose. For example, a sender can choose a route so that its datagram does not travel through a competitor's network.

If a datagram specifies a strict source route, all the routers defined in the option must be visited by the datagram. A router must not be visited if its IPv4 address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued.

Loose Source Route

A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

Timestamp

A timestamp option is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time. Knowing the time a datagram is processed can help users and managers track the behavior of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say *estimate* because, although all routers may use Universal time, their local clocks may not be synchronized.

20.3 IPv6

The network layer protocol in the TCPIIP protocol suite is currently IPv4 (Internetworking Protocol, version 4). IPv4 provides the host-to-host communication between systems in the Internet. Although IPv4 is well designed, data communication has evolved since the inception of IPv4 in the 1970s. IPv4 has some deficiencies (listed below) that make it unsuitable for the fast-growing Internet.

- Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
- The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
- The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided by IPv4.

To overcome these deficiencies, IPv6 (Internetworking Protocol, version 6), also known as IPng (Internetworking Protocol, next generation), was proposed and is now a standard. In IPv6, the Internet protocol was extensively modified to accommodate the unforeseen growth of the Internet. The format and the length of the IP address were changed along with the packet format. Related protocols, such as ICMP, were also modified. Other protocols in the network layer, such as ARP, RARP, and IGMP, were

either deleted or included in the ICMPv6 protocol (see Chapter 21). Routing protocols, such as RIP and OSPF (see Chapter 22), were also slightly modified to accommodate these changes. Communications experts predict that IPv6 and its related protocols will soon replace the current IP version. In this section first we discuss IPv6. Then we explore the strategies used for the transition from version 4 to version 6.

The adoption of IPv6 has been slow. The reason is that the original motivation for its development, depletion of IPv4 addresses, has been remedied by short-term strategies such as classless addressing and NAT. However, the fast-spreading use of the Internet, and new services such as mobile IP, IP telephony, and IP-capable mobile telephony, may eventually require the total replacement of IPv4 with IPv6.

Advantages

The next-generation IP, or IPv6, has some advantages over IPv4 that can be summarized as follows:

- Larger address space. An IPv6 address is 128 bits long, as we discussed in Chapter 19. Compared with the 32-bit address of IPv4, this is a huge (2^{96}) increase in the address space.
- Better header format. IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
- New options. IPv6 has new options to allow for additional functionalities.
- Allowance for extension. IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- Support for resource allocation. In IPv6, the type-of-service field has been removed, but a mechanism (calledjlow *label)* has been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- Support for more security. The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format

The IPv6 packet is shown in Figure 20.15. Each packet is composed of a mandatory base header followed by the payload. The payload consists of two parts: optional extension headers and data from an upper layer. The base header occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.

Base Header

Figure 20.16 shows the base header with its eight fields.

These fields are as follows:

- Version. This 4-bit field defines the version number of the IP. For IPv6, the value is 6.
- Priority. The 4-bit priority field defines the priority of the packet with respect to traffic congestion. We will discuss this field later.

Figure 20.15 *IPv6 datagram header and payload*

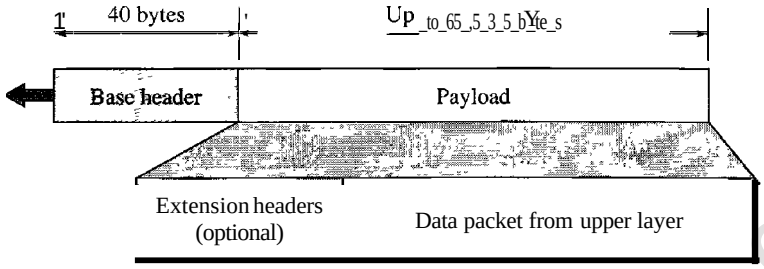
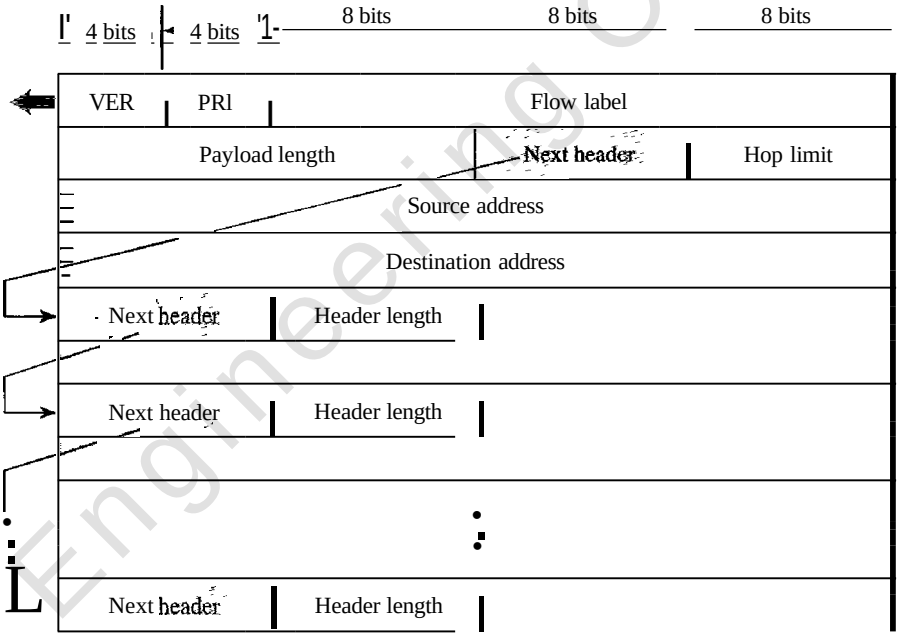


Figure 20.16 *Format of an IPv6 datagram*



- Flow label. The flow label is a 3-byte (24-bit) field that is designed to provide special handling for a particular flow of data. We will discuss this field later.
- Payload length. The 2-byte payload length field defines the length of the IP datagram excluding the base header.
- Next header. The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either one of the optional extension headers used by IP or the header of an encapsulated packet such as UDP or TCP. Each extension header also contains this field. Table 20.6 shows the values of next headers. Note that this field in version 4 is called the *protocol.*
- Hop limit. This 8-bit hop limit field serves the same purpose as the TIL field in IPv4.
- Source address. The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.

Table 20.6 *Next header codes for IPv6*

Code	*Next Header*
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

- **Destination address.** The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram. However, if source routing is used, this field contains the address of the next router.

Priority

The priority field of the IPv6 packet defines the priority of each packet with respect to other packets from the same source. For example, if one of two consecutive datagrams must be discarded due to congestion, the datagram with the lower **packet priority** will be discarded. IPv6 divides traffic into two broad categories: congestion-controlled and noncongestion-controlled.

Congestion-Controlled Traffic If a source adapts itself to traffic slowdown when there is congestion, the traffic is referred to as **congestion-controlled traffic.** For example, TCP, which uses the sliding window protocol, can easily respond to traffic. **In** congestion-controlled traffic, it is understood that packets may arrive delayed, lost, or out of order. Congestion-controlled data are assigned priorities from 0 to 7, as listed in Table 20.7. A priority of 0 is the lowest; a priority of 7 is the highest.

Table 20.7 *Priorities for congestion-controlled traffic*

Priority	*Meaning*
0	No specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

The priority descriptions are as follows:

- No specific traffic. A priority of 0 is assigned to a packet when the process does not define a priority.
- Background data. This group (priority 1) defines data that are usually delivered in the background. Delivery of the news is a good example.
- Unattended data traffic. If the user is not waiting (attending) for the data to be received, the packet will be given a priority of 2. E-mail belongs to this group. The recipient of an e-mail does not know when a message has arrived. In addition, an e-mail is usually stored before it is forwarded. A little bit of delay is of little consequence.
- Attended **bulk** data traffic. A protocol that transfers data while the user is waiting (attending) to receive the data (possibly with delay) is given a priority of 4. FTP and HTTP belong to this group.
- Interactive traffic. Protocols such as TELNET that need user interaction are assigned the second-highest priority (6) in this group.
- Control traffic. Control traffic is given the highest priority (7). Routing protocols such as OSPF and RIP and management protocols such as SNMP have this priority.

Noncongestion-Controlled Traffic This refers to a type of traffic that expects minimum delay. Discarding of packets is not desirable. Retransmission in most cases is impossible. In other words, the source does not adapt itself to congestion. Real-time audio and video are examples of this type of traffic.

Priority numbers from 8 to 15 are assigned to noncongestion-controlled traffic. Although there are not yet any particular standard assignments for this type of data, the priorities are usually based on how much the quality of received data is affected by the discarding of packets. Data containing less redundancy (such as low-fidelity audio or video) can be given a higher priority (15). Data containing more redundancy (such as high-fidelity audio or video) are given a lower priority (8). See Table 20.8.

Table 20.8 *Priorities for noncongestion-controlled traffic*

Priority	*Meaning*
8	Data with greatest redundancy
. . .	. . .
15	Data with least redundancy

Flow Label

A sequence of packets, sent from a particular source to a particular destination, that needs special handling by routers is called a *flow* of packets. The combination of the source address and the value of the *flow label* uniquely defines a flow of packets.

To a router, a flow is a sequence of packets that share the same characteristics, such as traveling the same path, using the same resources, having the same kind of security, and so on. A router that supports the handling of flow labels has a flow label table. The table has an entry for each active flow label; each entry defines the services required by

the corresponding flow label. When the router receives a packet, it consults its flow label table to find the corresponding entry for the flow label value defined in the packet. It then provides the packet with the services mentioned in the entry. However, note that the flow label itself does not provide the information for the entries of the flow label table; the information is provided by other means such as the hop-by-hop options or other protocols.

In its simplest form, a flow label can be used to speed up the processing of a packet by a router. When a router receives a packet, instead of consulting the routing table and going through a routing algorithm to define the address of the next hop, it can easily look in a flow label table for the next hop.

In its more sophisticated form, a flow label can be used to support the transmission of real-time audio and video. Real-time audio or video, particularly in digital form, requires resources such as high bandwidth, large buffers, long processing time, and so on. A process can make a reservation for these resources beforehand to guarantee that real-time data will not be delayed due to a lack of resources. The use of real-time data and the reservation of these resources require other protocols such as Real-Time Protocol (RTP) and Resource Reservation Protocol (RSVP) in addition to IPv6.

To allow the effective use of flow labels, three rules have been defined:

1. The flow label is assigned to a packet by the source host. The label is a random number between 1 and $2^{24} - 1$. A source must not reuse a flow label for a new flow while the existing flow is still active.
2. If a host does not support the flow label, it sets this field to zero. If a router does not support the flow label, it simply ignores it.
3. All packets belonging to the same flow have the same source, same destination, same priority, and same options.

Comparison Between IPv4 and IPv6 Headers

Table 20.9 compares IPv4 and IPv6 headers.

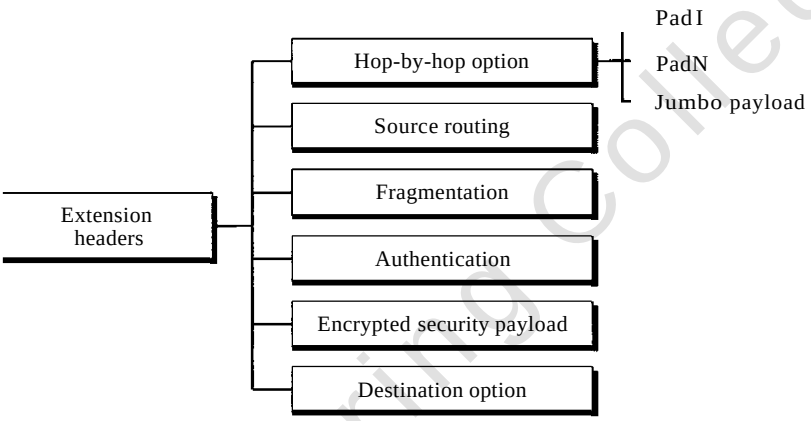
Table 20.9 *Comparison between IPv4 and IPv6 packet headers*

Comparison
1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Extension Headers

The length of the base header is fixed at 40 bytes. However, to give greater functionality to the IP datagram, the base header can be followed by up to six extension headers. Many of these headers are options in IPv4. Six types of extension headers have been defined, as shown in Figure 20.17.

Figure 20.17 *Extension header types*



Hop-by-Hop Option

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram. So far, only three options have been defined: Padl, PadN, and jumbo payload. The Padl option is 1 byte long and is designed for alignment purposes. PadN is similar in concept to Padi. The difference is that PadN is used when 2 or more bytes is needed for alignment. The jumbo payload option is used to define a payload longer than 65,535 bytes.

Source Routing The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4.

Fragmentation

The concept of fragmentation is the same as that in IPv4. However, the place where fragmentation occurs differs. In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels. In IPv6, only the original source can fragment. A source must use a path MTU discovery technique to find the smallest MTU supported by any network on the path. The source then fragments using this knowledge.

Authentication

The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data. We discuss this extension header when we discuss network security in Chapter 31.

Encrypted Security Payload

The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping. We discuss this extension header in Chapter 31.

Destination Option The destination option is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information.

Comparison Between IPv4 Options and IPv6 Extension Headers

Table 20.10 compares the options in IPv4 with the extension headers in IPv6.

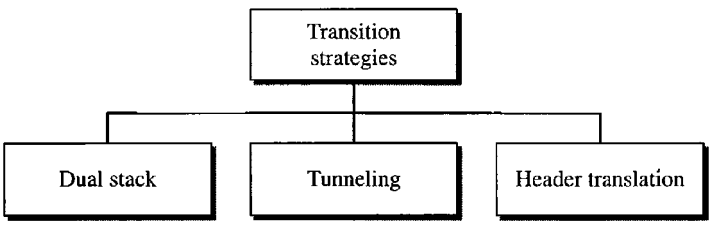
Table 20.10 *Comparison between IPv4 options and IPv6 extension headers*

Comparison
1. The no-operation and end-of-option options in IPv4 are replaced by Padl and PadN options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header in IPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
6. The authentication extension header is new in IPv6.
7. The encrypted security payload extension header is new in IPv6.

20.4 TRANSITION FROM IPv4 TO IPv6

Because of the huge number of systems on the Internet, the transition from IPv4 to IPv6 cannot happen suddenly. It takes a considerable amount of time before every system in the Internet can move from IPv4 to IPv6. The transition must be smooth to prevent any problems between IPv4 and IPv6 systems. Three strategies have been devised by the !E1F to help the transition (see Figure 20.18).

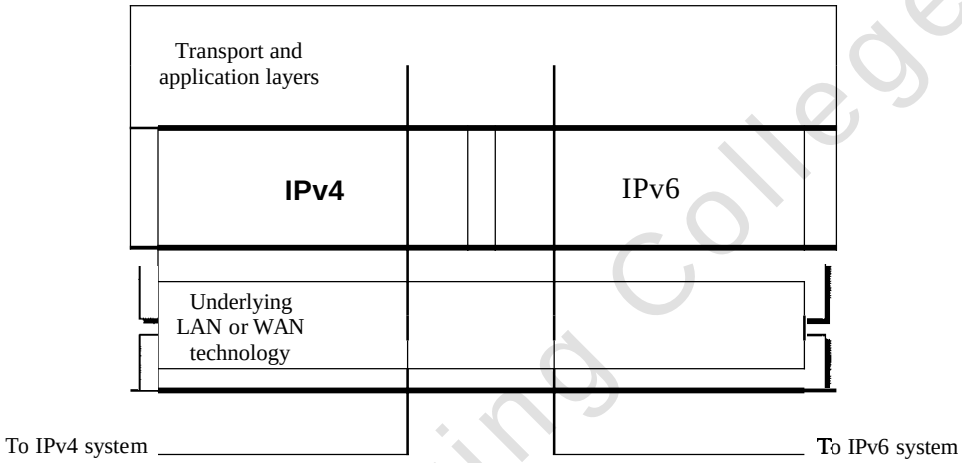
Figure 20.18 *Three transition strategies*



Dual Stack

It is recommended that all hosts, before migrating completely to version 6, have a **dual** stack of protocols. In other words, a station must run IPv4 and IPv6 simultaneously until all the Internet uses IPv6. See Figure 20.19 for the layout of a dual-stack configuration.

Figure 20.19 *Dual stack*

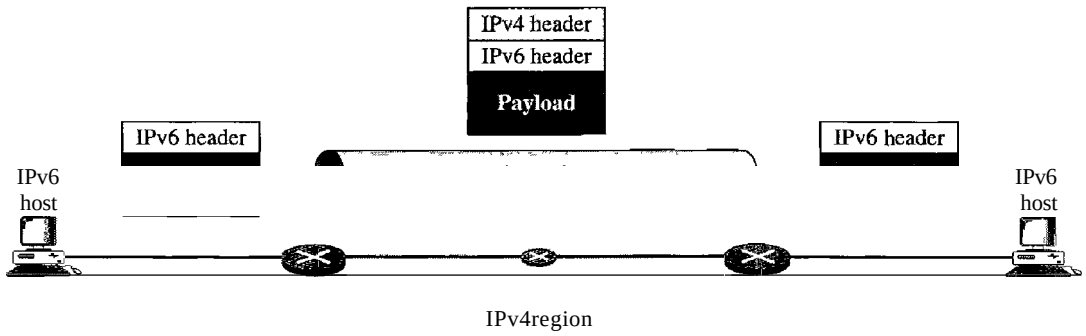


To determine which version to use when sending a packet to a destination, the source host queries the DNS. If the DNS returns an IPv4 address, the source host sends an IPv4 packet. If the DNS returns an IPv6 address, the source host sends an IPv6 packet.

Tunneling

Thnneling is a strategy used when two computers using IPv6 want to communicate with each other and the packet must pass through a region that uses IPv4. To pass through this region, the packet must have an IPv4 address. So the IPv6 packet is encapsulated in an IPv4 packet when it enters the region, and it leaves its capsule when it exits the region. It seems as if the IPv6 packet goes through a tunnel at one end and emerges at the other end. To make it clear that the IPv4 packet is carrying an IPv6 packet as data, the protocol value is set to 41. Tunneling is shown in Figure 20.20.

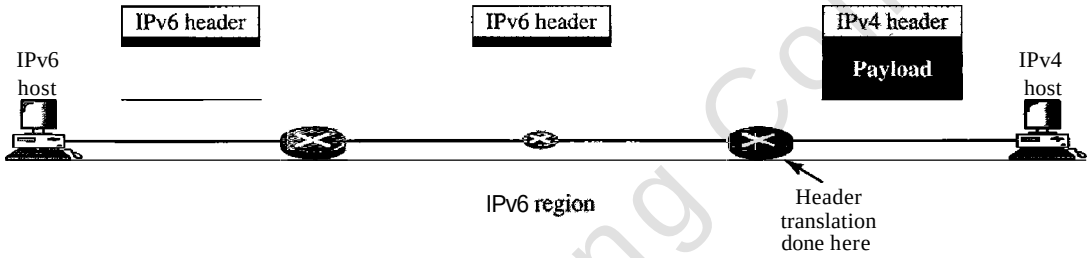
Figure 20.20 *Tunneling strategy*



Header Translation

Header translation is necessary when the majority of the Internet has moved to IPv6 but some systems still use IPv4. The sender wants to use IPv6, but the receiver does not understand IPv6. Tunneling does not work in this situation because the packet must be in the IPv4 format to be understood by the receiver. In this case, the header format must be totally changed through header translation. The header of the IPv6 packet is converted to an IPv4 header (see Figure 20.21).

Figure 20.21 *Header translation strategy*



Header translation uses the mapped address to translate an IPv6 address to an IPv4 address. Table 20.11 lists some rules used in transforming an IPv6 packet header to an IPv4 packet header.

Table 20.11 *Header translation*

Header Translation Procedure
1. The IPv6 mapped address is changed to an IPv4 address by extracting the rightmost 32 bits.
2. The value of the IPv6 priority field is discarded.
3. The type of service field in IPv4 is set to zero.
4. The checksum for IPv4 is calculated and inserted in the corresponding field.
5. The IPv6 flow label is ignored.
6. Compatible extension headers are converted to options and inserted in the IPv4 header. Some may have to be dropped.
7. The length of IPv4 header is calculated and inserted into the corresponding field.
8. The total length of the IPv4 packet is calculated and inserted in the corresponding field.

20.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

IPv4 is discussed in Chapter 8 of [For06], Chapter 3 of [Ste94J, Section 4.1 of [PD03], Chapter 18 of [Sta04], and Section 5.6 of [Tan03]. IPv6 is discussed in Chapter 27 of [For06] and [Los04].

Sites

- www.ietforg/rfc.html Information about RFCs

RFCs

A discussion of IPv4 can be found in following RFCs:

760,781,791,815,1025,1063,1071,1141,1190, 1191, 1624,2113

A discussion of IPv6 can be found in the following RFCs:

1365,1550,1678,1680,1682,1683,1686,1688,1726, 1752, 1826, 1883, 1884, 1886, 1887, 1955,2080,2373,2452,2463,2465,2466,2472,2492,2545,2590

20.6 KEY TERMS

authentication
base header
best-effort delivery
codepoint
connectionless service
connection-oriented service
datagram
destination address
destination option
differentiated services
dual stack
encrypted security payload (ESP)
end-of-option option
extension header
flow label
fragmentation
fragmentation offset
header length
header translation
hop limit
hop-by-hop option
Internet Protocol (IP)
Internet Protocol, next generation (IPng)
Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)
jumbo payload
loose source route option
maximum transfer unit (MTU)
next header
noncongestion-controlled traffic
no-operation option
packet priority
Padl
PadN
path MTU discovery technique
precedence

record route option
service type
source address
strict source route option
time to live
timestamp option
tunneling
type of service (TOS)

20.7 SUMMARY

- IPv4 is an unreliable connectionless protocol responsible for source-to-destination delivery.
- Packets in the IPv4 layer are called datagrams. A datagram consists of a header (20 to 60 bytes) and data. The maximum length of a datagram is 65,535 bytes.
- The MTU is the maximum number of bytes that a data link protocol can encapsulate. MTUs vary from protocol to protocol.
- Fragmentation is the division of a datagram into smaller units to accommodate the MTU of a data link protocol.
- The IPv4 datagram header consists of a fixed, 20-byte section and a variable options section with a maximum of 40 bytes.
- The options section of the IPv4 header is used for network testing and debugging.
- The six IPv4 options each have a specific function. They are as follows: filler between options for alignment purposes, padding, recording the route the datagram takes, selection of a mandatory route by the sender, selection of certain routers that must be visited, and recording of processing times at routers.
- IPv6, the latest version of the Internet Protocol, has a 128-bit address space, a revised header format, new options, an allowance for extension, support for resource allocation, and increased security measures.
- An IPv6 datagram is composed of a base header and a payload.
- Extension headers add functionality to the IPv6 datagram.
- Three strategies used to handle the transition from version 4 to version 6 are dual stack, tunneling, and header translation.

20.8 PRACTICE SET

Review Questions

1. What is the difference between the delivery of a frame in the data link layer and the delivery of a packet in the network layer?
2. What is the difference between connectionless and connection-oriented services? Which type of service is provided by IPv4? Which type of service is provided by IPv6?
3. Define fragmentation and explain why the IPv4 and IPv6 protocols need to fragment some packets. Is there any difference between the two protocols in this matter?

4. Explain the procedure for checksum calculation and verification in the IPv4 protocol. What part of an IPv4 packet is covered in the checksum calculation? Why? Are options, if present, included in the calculation?
5. Explain the need for options in IPv4 and list the options mentioned in this chapter with a brief description of each.
6. Compare and contrast the fields in the main headers of IPv4 and IPv6. Make a table that shows the presence or absence of each field.
7. Both IPv4 and IPv6 assume that packets may have different priorities or precedences. Explain how each protocol handles this issue.
8. Compare and contrast the options in IPv4 and the extension headers in IPv6. Make a table that shows the presence or absence of each.
9. Explain the reason for the elimination of the checksum in the IPv6 header.
10. List three transition strategies to move from IPv4 to IPv6. Explain the difference between tunneling and dual stack strategies during the transition period. When is each strategy used?

Exercises

11. Which fields of the IPv4 header change from router to router?
12. Calculate the HLEN (in IPv4) value if the total length is 1200 bytes, 1176 of which is data from the upper layer.
13. Table 20.5 lists the MTUs for many different protocols. The MTUs range from 296 to 65,535. What would be the advantages of having a large MTU? What would be the advantages of having a small MTU?
14. Given a fragmented datagram (in IPv4) with an offset of 120, how can you determine the first and last byte numbers?
15. Can the value of the header length in an IPv4 packet be less than 5? When is it exactly 5?
16. The value of HLEN in an IPv4 datagram is 7. How many option bytes are present?
17. The size of the option field of an IPv4 datagram is 20 bytes. What is the value of HLEN? What is the value in binary?
18. The value of the total length field in an IPv4 datagram is 36, and the value of the header length field is 5. How many bytes of data is the packet carrying?
19. An IPv4 datagram is carrying 1024 bytes of data. If there is no option information, what is the value of the header length field? What is the value of the total length field?
20. A host is sending 100 datagrams to another host. If the identification number of the first datagram is 1024, what is the identification number of the last (in IPv4)?
21. An IPv4 datagram arrives with fragmentation offset of 0 and an *M* bit *(more* fragment bit) of 0. Is this a first fragment, middle fragment, or last fragment?
22. An IPv4 fragment has arrived with an offset value of 100. How many bytes of data were originally sent by the source before the data in this fragment?

23. An IPv4 datagram has arrived with the following information **in** the header (in hexadecimal):

    Ox45 00 00 54 00 03 58 50 20 06 00 00 7C 4E 03 02 B4 OE OF 02

    a. Is the packet corrupted?
    b. Are there any options?
    c. Is the packet fragmented?
    d. What is the size of the data?
    e. How many more routers can the packet travel to?
    f. What is the identification number of the packet?
    g. What is the type of service?

24. In an IPv4 datagram, the *M* bit is 0, the value of HLEN is 5, the value of total length is 200, and the offset value is 200. What is the number of the first byte and number of the last byte **in** this datagram? Is this the last fragment, the first fragment, or a middle fragment?

Research Activities

25. Find out why there are two security protocols (AH and ESP) **in** IPv6.

CHAPTER 21

Network Layer: Address Mapping, Error Reporting, and Multicasting

In Chapter 20 we discussed the Internet Protocol (IP) as the main protocol at the network layer. IP was designed as a best-effort delivery protocol, but it lacks some features such as flow control and error control. It is a host-to-host protocol using logical addressing. To make IP more responsive to some requirements in today's intemetworking, we need the help of other protocols.

We need protocols to create a mapping between physical and logical addresses. IP packets use logical (host-to-host) addresses. These packets, however, need to be encapsulated in a frame, which needs physical addresses (node-to-node). We will see that a protocol called ARP, the Address Resolution Protocol, is designed for this purpose. We sometimes need reverse mapping-mapping a physical address to a logical address. For example, when booting a diskless network or leasing an IP address to a host. Three protocols are designed for this purpose: RARP, BOOTp, and DHCP.

Lack of flow and error control in the Internet Protocol has resulted in another protocol, ICMP, that provides alerts. It reports congestion and some types of errors in the network or destination host.

IP was originally designed for unicast delivery, one source to one destination. As the Internet has evolved, the need for multicast delivery, one source to many destinations, has increased tremendously. IGMP gives IP a multicast capability.

In this chapter, we discuss the protocols ARP, RARP, BOOTP, DHCP, and IGMP in some detail. We also discuss ICMPv6, which will be operational when IPv6 is operational. ICMPv6 combines ARP, ICMP, and IGMP in one protocol.

21.1 ADDRESS MAPPING

An internet is made of a combination of physical networks connected by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host. The hosts and routers are recognized at the network level by their logical (IP) addresses.

However, packets pass through physical networks to reach these hosts and routers. At the physical level, the hosts and routers are recognized by their physical addresses.

A physical address is a local address. Its jurisdiction is a local network. It must be unique locally, but is not necessarily unique universally. It is called a *physical* address because it is usually (but not always) implemented in hardware. An example of a physical address is the 48-bit MAC address in the Ethernet protocol, which is imprinted on the NIC installed in the host or router.

The physical address and the logical address are two different identifiers. We need both because a physical network such as Ethernet can have two different protocols at the network layer such as IP and IPX (Novell) at the same time. Likewise, a packet at a network layer such as IP may pass through different physical networks such as Ethernet and LocalTalk (Apple).

This means that delivery of a packet to a host or a router requires two levels of addressing: logical and physical. We need to be able to map a logical address to its corresponding physical address and vice versa. These can be done by using either static or dynamic mapping.

Static mapping involves in the creation of a table that associates a logical address with a physical address. This table is stored in each machine on the network. Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table. This has some limitations because physical addresses may change in the following ways:

1. A machine could change its NIC, resulting in a new physical address.
2. In some LANs, such as LocalTalk, the physical address changes every time the computer is turned on.
3. A mobile computer can move from one physical network to another, resulting in a change in its physical address.

To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

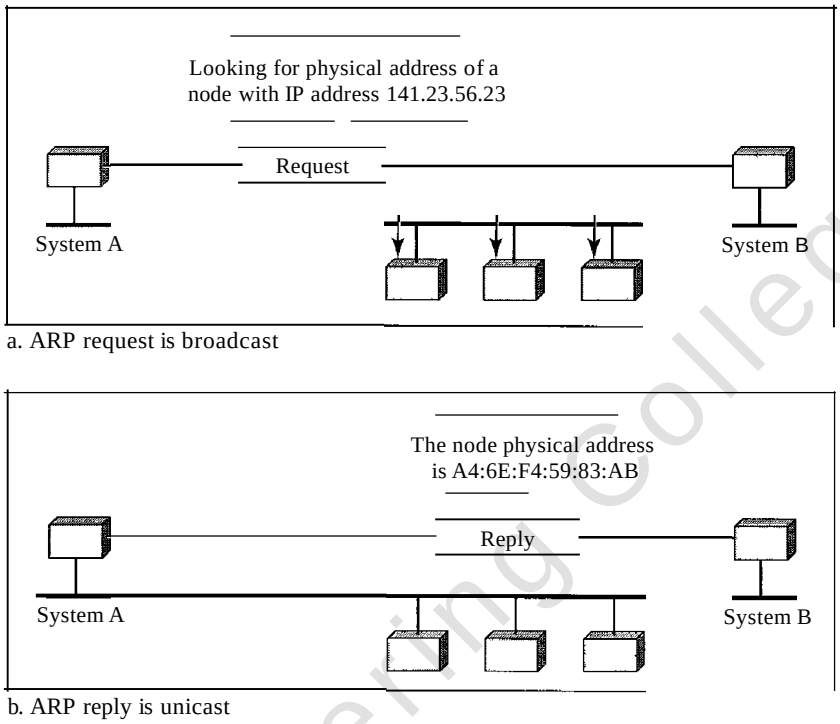
In dynamic mapping each time a machine knows one of the two addresses (logical or physical), it can use a protocol to find the other one.

Mapping Logical to Physical Address: ARP

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver. The logical (IP) address is obtained from the DNS (see Chapter 25) if the sender is the host or it is found in a routing table (see Chapter 22) if the sender is a router. But the IP datagram must be encapsulated in a frame to be able to pass through the physical network. This means that the sender needs the physical address of the receiver. The host or the router sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the physical address of the receiver, the query is broadcast over the network (see Figure 21.1).

Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and physical addresses. The packet is unicast directly to the inquirer by using the physical address received in the query packet.

Figure 21.1 *ARP operation*



In Figure 21.1a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address 141.23.56.23. System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IF address of 141.23.56.23.

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 21.1b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination by using the physical address it received.

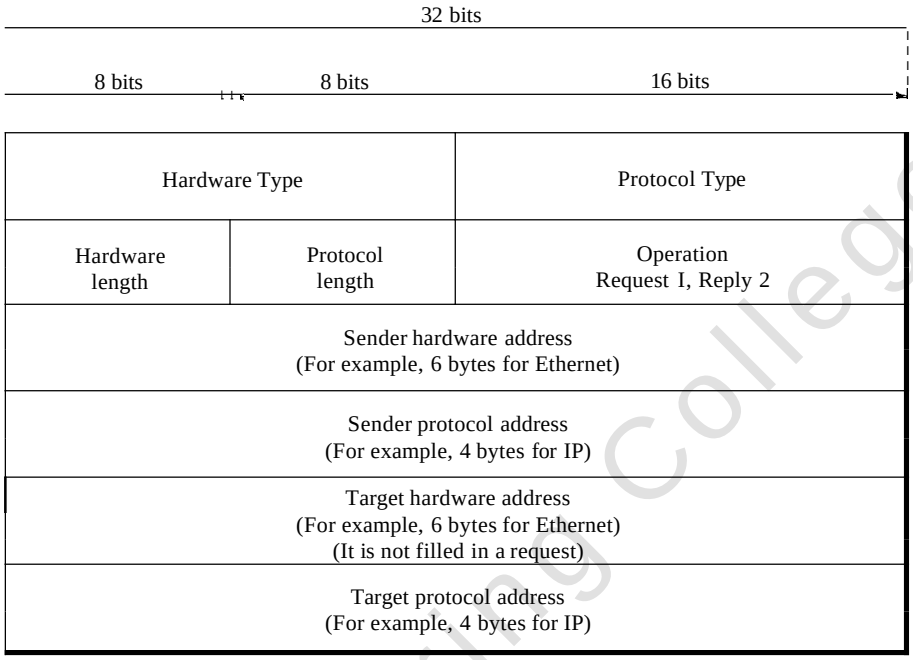
Cache Memory

Using ARP is inefficient if system A needs to broadcast an ARP request for each IP packet it needs to send to system B. It could have broadcast the IP packet itself. ARP can be useful if the ARP reply is cached (kept in cache memory for a while) because a system normally sends several packets to the same destination. A system that receives an ARP reply stores the mapping in the cache memory and keeps it for 20 to 30 minutes unless the space in the cache is exhausted. Before sending an ARP request, the system first checks its cache to see if it can find the mapping.

Packet Format

Figure 21.2 shows the format of an ARP packet.

Figure 21.2 *ARP packet*



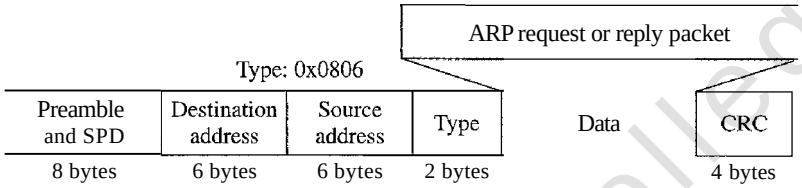
Thc fields arc as follows:

- Hardware type. This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For example, Ethernet is given type 1. ARP can be used on any physical network.
- Protocol type. This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 0800_{16}, ARP can be used with any higher-level protocol.
- Hardware length. This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.
- Protocol length. This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.
- Operation. This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1) andARP reply (2).
- Sender hardware address. This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.
- Sender protocol address. This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.
- Target hardware address. This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is alIOs because the sender does not know the physical address of the target.
- Target protocol address. This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

Encapsulation

An ARP packet is encapsulated directly into a data link frame. For example, in Figure 21.3 an ARP packet is encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame are an ARP packet.

Figure 21.3 *Encapsulation of ARP packet*



Operation

Let us see how ARP functions on a typical internet. First we describe the steps involved. Then we discuss the four cases in which a host or router needs to use ARP. These are the steps involved in an ARP process:

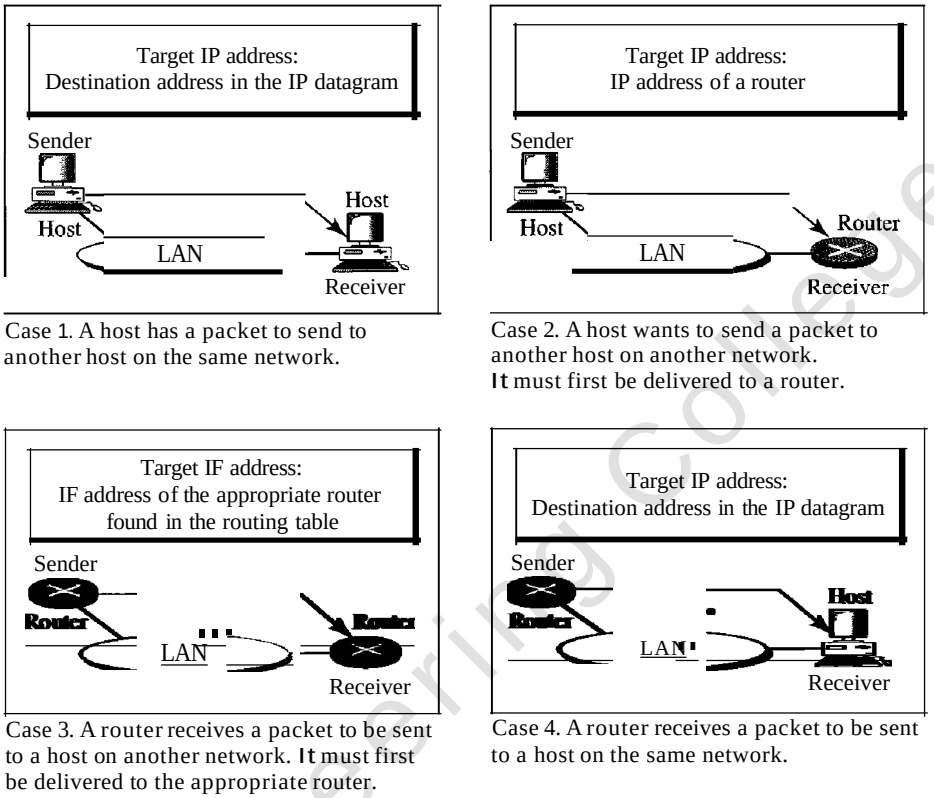
1. The sender knows the IP address of the target. We will see how the sender obtains this shortly.
2. IP asks ARP to create an ARP request message, filling in the sender physical address, the sender IP address, and the target IP address. The target physical address field is filled with 0s.
3. The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.
4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.
5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.
6. The sender receives the reply message. It now knows the physical address of the target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

Four Different Cases

The following are four different cases in which the services of ARP can be used (see Figure 21.4).

1. The sender is a host and wants to send a packet to another host on the same network. In this case, the logical address that must be mapped to a physical address is the destination IP address in the datagram header.

Figure 21.4 *Four cases using ARP*



Case 1. A host has a packet to send to another host on the same network.

Case 2. A host wants to send a packet to another host on another network. It must first be delivered to a router.

Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.

Case 4. A router receives a packet to be sent to a host on the same network.

2. The sender is a host and wants to send a packet to another host on another network. In this case, the host looks at its routing table and finds the IP address of the next hop (router) for this destination. If it does not have a routing table, it looks for the IP address of the default router. The IP address of the router becomes the logical address that must be mapped to a physical address.
3. The sender is a router that has received a datagram destined for a host on another network. It checks its routing table and finds the IP address of the next router. The IP address of the next router becomes the logical address that must be mapped to a physical address.
4. The sender is a router that has received a datagram destined for a host on the same network. The destination IP address of the datagram becomes the logical address that must be mapped to a physical address.

An ARP request is broadcast; an ARP reply is unicast.

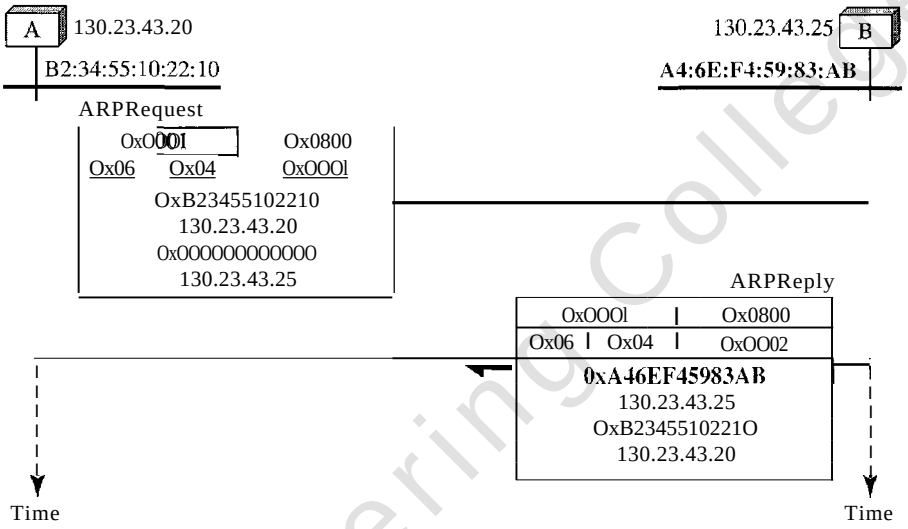
Example 21.1

A host with IP address 130.23.43.20 and physical address B2:34:55: 10:22: 10 has a packet to send to another host with IP address 130.23.43.25 and physical address A4:6E:F4:59:83:AB (which is unknown to the first host). The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

Solution

Figure 21.5 shows the ARP request and reply packets. Note that the ARP data field in this case is 28 bytes, and that the individual addresses do not fit in the 4-byte boundary. That is why we do not show the regular 4-byte boundaries for these addresses.

Figure 21.5 *Example* 21.1, *anARP request and reply*

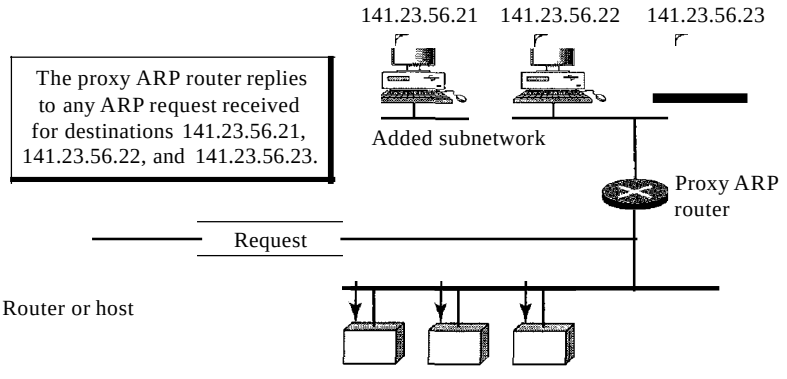


ProxyARP

A technique called *proxy* ARP is used to create a subnetting effect. A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router.

Let us give an example. In Figure 21.6 the ARP installed on the right-hand host will answer only to anARP request with a target IP address of 141.23.56.23.

Figure 21.6 *Proxy ARP*



However, the administrator may need to create a subnet without changing the whole system to recognize subnetted addresses. One solution is to add a router running a proxy ARP. In this case, the router acts on behalf of all the hosts installed on the subnet. When it receives an ARP request with a target IP address that matches the address of one of its protégés (141.23.56.21, 141.23.56.22, or 141.23.56.23), it sends an ARP reply and announces its hardware address as the target hardware address. When the router receives the IP packet, it sends the packet to the appropriate host.

Mapping Physical to Logical Address: RARP, BOOTP, and DHCP

There are occasions in which a host knows its physical address, but needs to know its logical address. This may happen in two cases:

1. A diskless station is just booted. The station can find its physical address by checking its interface, but it does not know its IP address.
2. An organization does not have enough IP addresses to assign to each station; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

RARP

Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address. Each host or router is assigned one or more logical (IP) addresses, which are unique and independent of the physical (hardware) address of the machine. To create an IP datagram, a host or a router needs to know its own IP address or addresses. The IP address of a machine is usually read from its configuration file stored on a disk file.

However, a diskless machine is usually booted from ROM, which has minimum booting information. The ROM is installed by the manufacturer. It cannot include the IP address because the IP addresses on a network are assigned by the network administrator.

The machine can get its physical address (by reading its NIC, for example), which is unique locally. It can then use the physical address to get the logical address by using the RARP protocol. A RARP request is created and broadcast on the local network. Another machine on the local network that knows all the IP addresses will respond with a RARP reply. The requesting machine must be running a RARP client program; the responding machine must be running a RARP server program.

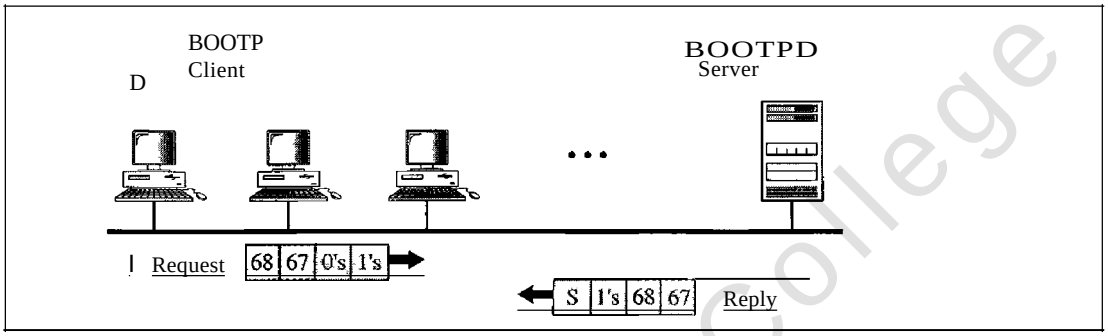
There is a serious problem with RARP: Broadcasting is done at the data link layer. The physical broadcast address, all is in the case of Ethernet, does not pass the boundaries of a network. This means that if an administrator has several networks or several subnets, it needs to assign a RARP server for each network or subnet. This is the reason that RARP is almost obsolete. Two protocols, BOOTP and DHCp, are replacing RARP.

BOOTP

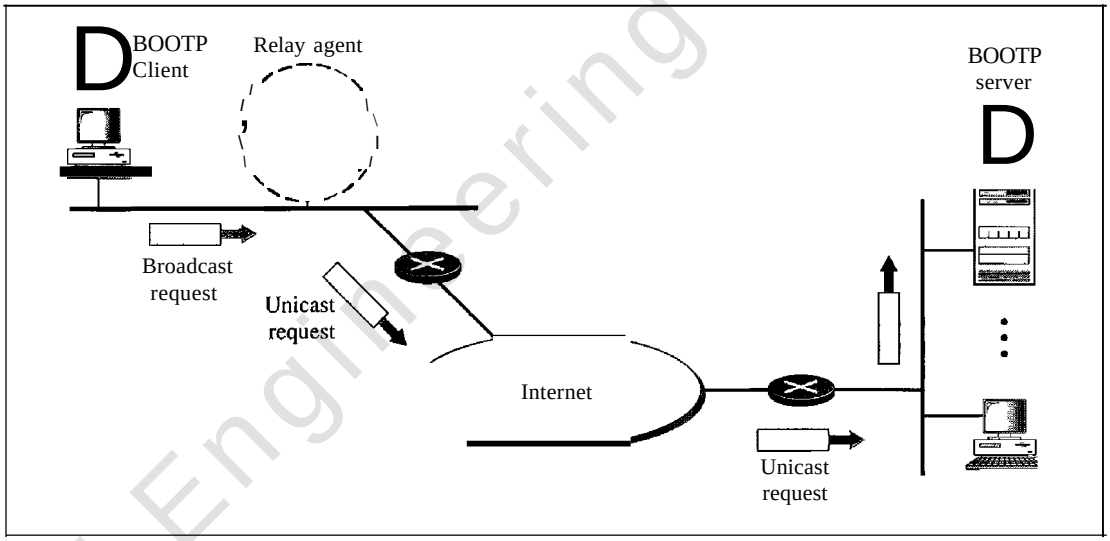
The Bootstrap Protocol (BOOTP) is a client/server protocol designed to provide physical address to logical address mapping. BOOTP is an application layer protocol. The administrator may put the client and the server on the same network or on different

networks, as shown in Figure 21.7. BOOTP messages are encapsulated in a UDP packet, and the UDP packet itself is encapsulated in an IP packet.

Figure 21.7 *BOOTP client and server on the same and different network*



a. Client and server on the same network



b. Client and server on different networks

The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client simply uses all as as the source address and allIs as the destination address.

One of the advantages of BOOTP over RARP is that the client and server are application-layer processes. As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks. However, there is one problem that must be solved. The BOOTP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this case is called a relay agent. The relay agent knows the unicast address of a BOOTP server. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the BOOTP server. The packet,

carrying a unicast destination address, is routed by any router and reaches the BOOTP server. The BOOTP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the BOOTP client.

DHCP

BOOTP is not a dynamic configuration protocol. When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address. This implies that the binding between the physical address and the IP address of the client already exists. The binding is predetermined.

However, what if a host moves from one physical network to another? What if a host wants a temporary IP address? BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator. BOOTP is a static configuration protocol.

The Dynamic Host Configuration Protocol (DHCP) has been devised to provide static and dynamic address allocation that can be manual or automatic.

DHCP provides static and dynamic address allocation that can be manual or automatic.

Static Address Allocation In this capacity DHCP acts as BOOTP does. It is backward-compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.

Dynamic Address Allocation DHCP has a second database with a pool of available IP addresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

When a DHCP client sends a request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned. On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.

The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network (as is a subscriber to a service provider). DHCP provides temporary IP addresses for a limited time.

The addresses assigned from the pool are temporary addresses. The DHCP server issues a lease for a specific time. When the lease expires, the client must either stop using the IP address or renew the lease. The server has the option to agree or disagree with the renewal. If the server disagrees, the client stops using the address.

Manual and Automatic Configuration One major problem with the BOOTP protocol is that the table mapping the IP addresses to physical addresses needs to be manually configured. This means that every time there is a change in a physical or IP address, the administrator needs to manually enter the changes. DHCP, on the other hand, allows both manual and automatic configurations. Static addresses are created manually; dynamic addresses are created automatically.

21.2 ICMP

As discussed in Chapter 20, the IP provides unreliable and connectionless datagram delivery. It was designed this way to make efficient use of network resources. The IP protocol is a best-effort delivery service that delivers a datagram from its original source to its final destination. However, it has two deficiencies: lack of error control and lack of assistance mechanisms.

The IP protocol has no error-reporting or error-correcting mechanism. What happens if something goes wrong? What happens if a router must discard a datagram because it cannot find a router to the final destination, or because the time-to-live field has a zero value? What happens if the final destination host must discard all fragments of a datagram because it has not received all fragments within a predetermined time limit? These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network administrator needs information from another host or router.

The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protoco1.

Types of Messages

ICMP messages are divided into two broad categories: error-reporting messages and query messages.

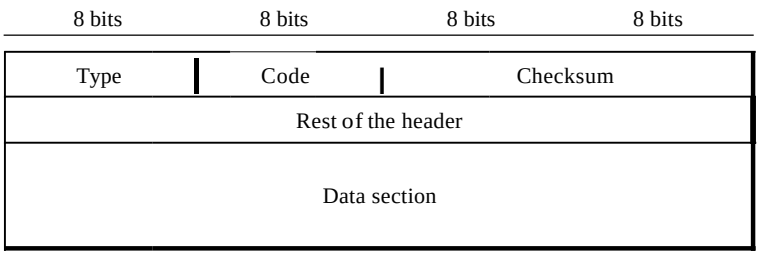
The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet.

The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network, and routers can help a node redirect its messages.

Message Format

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 21.8 shows, the first field, ICMP type, defines the type of the

Figure 21.8 *Generalformat of[CM? messages*



message. The code field specifies the reason for the particular message type. The last common field is the checksum field (to be discussed later in the chapter). The rest of the header is specific for each message type.

The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of the query.

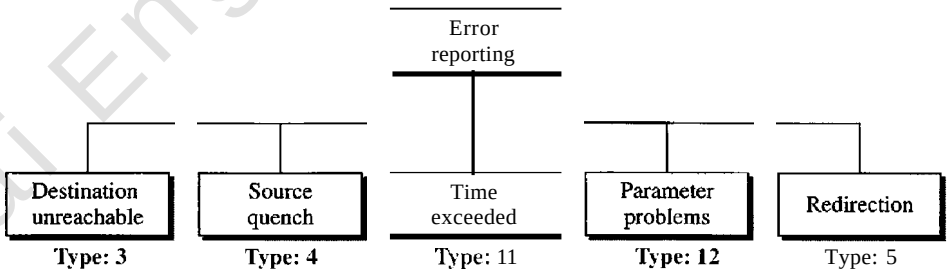
Error Reporting

One of the main responsibilities of ICMP is to report errors. Although technology has produced increasingly reliable transmission media, errors still exist and must be handled. IP, as discussed in Chapter 20, is an unreliable protocol. This means that error checking and error control are not a concern of IP. ICMP was designed, in part, to compensate for this shortcoming. However, ICMP does not correct errors-it simply reports them. Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses. ICMP uses the source IP address to send the error message to the source (originator) of the datagram.

ICMP always reports error messages to the original source.

Five types of errors are handled: destination unreachable, source quench, time exceeded, parameter problems, and redirection (see Figure 21.9).

Figure 21.9 *Error-reporting messages*

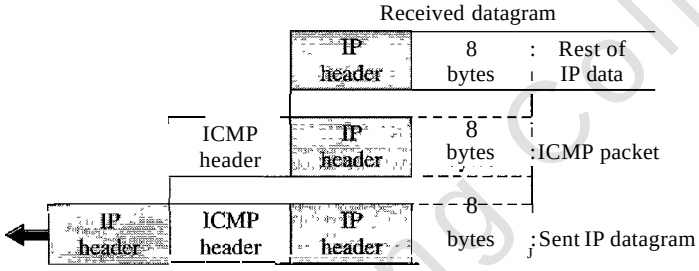


The following are important points about ICMP error messages:

- No ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- No ICMP error message will be generated for a fragmented datagram that is not the first fragment.
- No IeMP error message will be generated for a datagram having a multicast address.
- No ICMP error message will be generated for a datagram having a special address such as 127.0.0.0 or 0.0.0.0.

Note that all error messages contain a data section that includes the IP header of the original datagram plus the first 8 bytes of data in that datagram. The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because, as we will see in Chapter 23 on UDP and TCP protocols, the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error. ICMP forms an error packet, which is then encapsulated in an IP datagram (see Figure 21.10).

Figure 21.10 *Contents of data field for the error messages*



Destination Unreachable

When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a destination-unreachable message back to the source host that initiated the datagram. Note that destination-unreachable messages can be created by either a router or the destination host.

Source Quench

The IP protocol is a connectionless protocol. There is no communication between the source host, which produces the datagram, the routers, which forward it, and the destination host, which processes it. One of the ramifications of this absence of communication is the lack of *flow control.* IP does not have a flow control mechanism embedded in the protocol. The lack of flow control can create a major problem in the operation of IP: congestion. The source host never knows if the routers or the destination host has been overwhelmed with datagrams. The source host never knows if it is producing datagrams faster than can be forwarded by routers or processed by the destination host.

The lack of flow control can create congestion in routers or the destination host. A router or a host has a limited-size queue (buffer) for incoming datagrams waiting to be forwarded (in the case of a router) or to be processed (in the case of a host). If the datagrams are received much faster than they can be forwarded or processed, the queue may overflow. In this case, the router or the host has no choice but to discard some of the datagrams. The source-quench message in ICMP was designed to add a kind of flow control to the IP. When a router or host discards a datagram due to congestion, it sends a source-quench message to the sender of the datagram. This message has two purposes. First, it informs the source that the datagram has been discarded. Second, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process.

Time Exceeded

The time-exceeded message is generated in two cases: As we see in Chapter 22, routers use routing tables to find the next hop (next router) that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loop or a cycle, going from one router to the next or visiting a series of routers endlessly. As we saw in Chapter 20, each datagram contains a field called *time to live* that controls this situation. When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source. Second, a time-exceeded message is also generated when not all fragments that make up a message arrive at the destination host within a certain time limit.

Parameter Problem

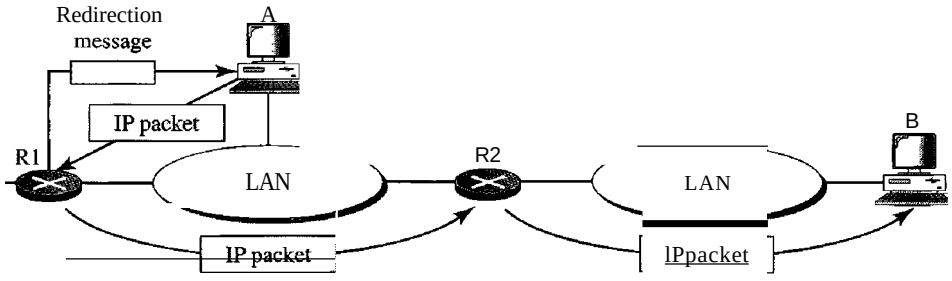
Any ambiguity in the header part of a datagram can Create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.

Redirection

When a router needs to send a packet destined for another network, it must know the IP address of the next appropriate router. The same is true if the sender is a host. Both routers and hosts, then, must have a routing table to find the address of the router or the next router. Routers take part in the routing update process, as we will see in Chapter 22, and are supposed to be updated constantly. Routing is dynamic.

However, for efficiency, hosts do not take part in the routing update process because there are many more hosts in an internet than routerS. Updating the routing tables of hosts dynamically produces unacceptable traffic. The hosts usually use static routing. When a host comes up, its routing table has a limited number of entries. It usually knows the IP address of only one router, the default router. For this reason, the host may send a datagram, which is destined for another network, to the wrong router. In this case, the router that receives the datagram will forward the datagram to the correct router. However, to update the routing table of the host, it sends a redirection message to the host. This concept of redirection is shown in Figure 21.11. Host A wants to send a datagram to host B.

Figure **21.11** *Redirection concept*



Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead. Router R1, after consulting its table, finds that the packet should have gone to R2. It sends the packet to R2 and, at the same time, sends a redirection message to host A. Host A's routing table can now be updated.

Query

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages, as shown in Figure 21.12. In this type of ICMP message, a node sends a message that is answered in a specific format by the destination node. A query message is encapsulated in an IP packet, which in tum is encapsulated in a data link layer frame. However, in this case, no bytes of the original IP are included in the message, as shown in Figure 21.13.

Figure 21.12 *Query messages*

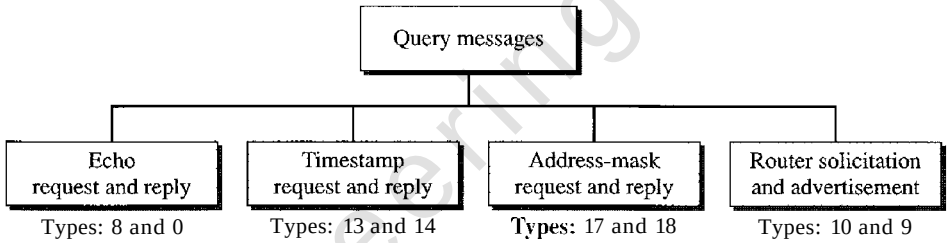
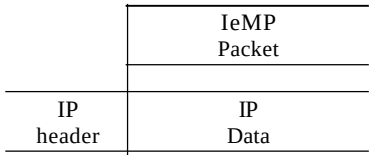


Figure 21.13 *Encapsulation of ICMP query messages*



Echo Request and Reply

The echo-request and echo-reply messages are designed for diagnostic purposes. Network managers and users utilize this pair of messages to identify network problems. The combination of echo-request and echo-reply messages detennines whether two systems (hosts or routers) can communicate with each other. The echo-request and echo-reply messages can be used to determine if there is communication at the IP level. Because ICMP messages are encapsulated in IP datagrams, the receipt of an echo-reply message by the machine that sent the echo request is proof that the IP protocols in the sender and receiver are communicating with each other using the IP datagram. Also, it is proof that the intermediate routers are receiving, processing, and forwarding IP datagrams. Today, most systems provide a version of the *ping* command that can create a series (instead of just one) of echo-request and echo-reply messages, providing statistical information. We will see the use of this program at the end of the chapter.

Timestamp Request and Reply

Two machines (hosts or routers) can use the timestamp request and timestamp reply messages to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines.

Address-Mask Request and Reply

A host may know its IP address, but it may not know the corresponding mask. For example, a host may know its IP address as 159.31.17.24, but it may not know that the corresponding mask is /24. To obtain its mask, a host sends an address-mask-request message to a router on the LAN. If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host. This can be applied to its full IP address to get its subnet address.

Router Solicitation and Advertisement

As we discussed in the redirection message section, a host that wants to send data to a host on another network needs to know the address of routers connected to its own network. Also, the host must know if the routers are alive and functioning. The router-solicitation and router-advertisement messages can help in this situation. A host can broadcast (or multicast) a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited. Note that when a router sends out an advertisement, it announces not only its own presence but also the presence of all routers on the network of which it is aware.

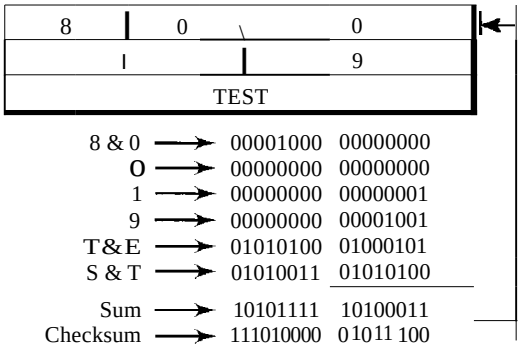
Checksum

In Chapter 10, we learned the concept and idea of the checksum. In ICMP the checksum is calculated over the entire message (header and data).

Example 21.2

Figure 21.14 shows an example of checksum calculation for a simple echo-request message. We randomly chose the identifier to be 1 and the sequence number to be 9. The message is divided

Figure 21.14 *Example of checksum calculation*



into 16-bit (2-byte) words. The words are added and the sum is complemented. Now the sender can put this value in the checksum field.

Debugging Tools

There are several tools that can be used in the Internet for debugging. We can determine the viability of a host or router. We can trace the route of a packet. We introduce two tools that use ICMP for debugging: *ping* and *traceroute*. We will introduce more tools in future chapters after we have discussed the corresponding protocols.

Ping

We can use the *ping* program to find if a host is alive and responding. We use *ping* here to see how it uses ICMP packets.

The source host sends ICMP echo-request messages (type: 8, code: 0); the destination, if alive, responds with ICMP echo-reply messages. The *ping* program sets the identifier field in the echo-request and echo-reply message and stans the sequence number from 0; this number is incremented by 1 each time a new message is sent. Note that *ping* can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

Example 21.3

We use the *ping* program to test the server fhda.edu. The result is shown below:

```
$ ping thda.edu
PING thda.edu (153.18.8.1)  56 (84)  bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=O    ttl=62    time=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=l    ttl=62    time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2    ttl=62    time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3    ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4    ttl=62    time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5    ttl=62    time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6    tt1=62    time=1.94 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=7   ttl=62    time=1.94 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=8   ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9    ttl=62    time=1.89 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=lO   ttl==62   time=1.98 ms

--- thda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103ms
    rtt minJavg/max = 1.899/1.955/2.041 ms
```

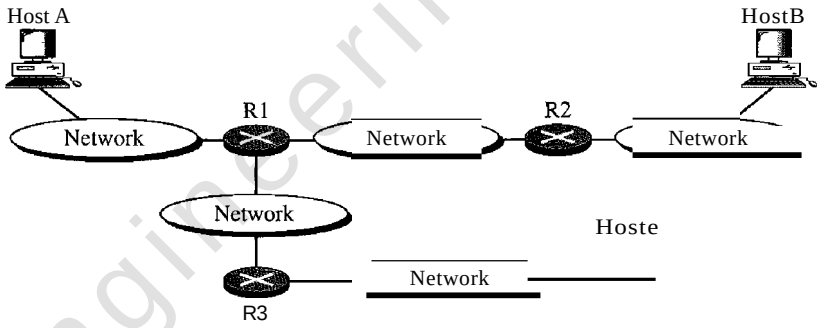
The *ping* program sends messages with sequence numbers starting from 0. For each probe it gives us the RTT time. The TTL (time to live) field in the IP datagram that encapsulates an ICMP message has been set to 62, which means the packet cannot travel mare than 62 hops. At the beginning, *ping* defines the number of data bytes as 56 and the total number of bytes as 84. It is obvious that if we add 8 bytes of ICMP header and 20 bytes of IP header to 56, the result is 84. However,

note that in each probe *ping* defines the number of bytes as 64. This is the total number of bytes in the ICMP packet (56 + 8). The *ping* program continues to send messages, if we do not stop it by using the interrupt key (ctrl + c, for example). After it is interrupted, it prints the statistics of the probes. It tells us the number of packets sent, the number of packets received, the total time, and the RTT minimum, maximum, and average. Some systems may print more infonnation.

Tracerollte

The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the route of a packet from the source to the destination. We have seen an application of the *traceroute* program to simulate the loose source route and strict source route options of an IP datagram in Chapter 20. We use this program in conjunction with ICMP packets in this chapter. The program elegantly uses two ICMP messages, time exceeded and destination unreachable, to find the route of a packet. This is a program at the application level that uses the services of UDP (see Chapter 23). Let us show the idea of the *traceroute* program by using Figure 21.15.

Figure 21.15 *The traceroute program operation*



Given the topology, we know that a packet from host A to host B travels through routers Rl and R2. However, most of the time, we are not aware of this topology. There could be several routes from A to B. The *traceroute* program uses the ICMP messages and the TTL (time to live) field in the IP packet to find the route.

1. The *traceroute* program uses the following steps to find the address of the router Rl and the round-trip time between host A and router Rl.
   a. The *traceroute* application at host A sends a packet to destination Busing UDP; the message is encapsulated in an IP packet with a TTL value of 1. The program notes the time the packet is sent.
   b. Router Rl receives the packet and decrements the value of TTL to 0. It then discards the packet (because TTL is 0). The router, however, sends a time-exceeded ICMP message (type: 11, code: 0) to show that the TTL value is 0 and the packet was discarded.
   c. The *traceroute* program receives the ICMP messages and uses the destination address of the IP packet encapsulating ICMP to find the IP address of router Rl. The program also makes note of the time the packet has arrived. The difference between this time and the time at step a is the round-trip time.

The *traceroute* program repeats steps a to c three times to get a better average round-trip time. The first trip time may be much longer than the second or third because it takes time for the ARP program to find the physical address of router RI. For the second and third trips, ARP has the address in its cache.

2. The *traceroute* program repeats the previous steps to find the address of router R2 and the round-trip time between host A and router R2. However, in this step, the value of TTL is set to 2. So router RI forwards the message, while router R2 discards it and sends a time-exceeded ICMP message.
3. The *traceroute* program repeats step 2 to find the address of host B and the round-trip time between host A and host B. When host B receives the packet, it decrements the value of TIL, but it does not discard the message since it has reached its final destination. How can an ICMP message be sent back to host A? The *traceroute* program uses a different strategy here. The destination port of the UDP packet is set to one that is not supported by the UDP protocol. When host B receives the packet, it cannot find an application program to accept the delivery. It discards the packet and sends an ICMP destination-unreachable message (type: 3, code: 3) to host A. Note that this situation does not happen at router RI or R2 because a router does not check the UDP header. The *traceroute* program records the destination address of the arrived IP datagram and makes note of the round-trip time. Receiving the destination-unreachable message with a code value 3 is an indication that the whole route has been found and there is no need to send more packets.

Example 21.4

We use the *traceroute* program to find the route from the computer voyager.deanza.edu to the server fhda.edu. The following shows the result:

```
$ traceroute fbda.edu
traceroute to fbda.edu  (153.18.8.1),30 hops max, 38 byte packets
 1 Dcore.fhda.edu       (153.18.31.254)   0.995 ms   0.899 ms   0.878 ms
 2 Dbackup.fhda.edu     (153.18.251.4)    1.039 fiS  1.064 fiS  1.083 fiS
 3 tiptoe.fhda.edu      (153.18.8.1)      1.797 illS 1.642 ms   1.757 fiS
```

The unnumbered line after the command shows that the destination is 153.18.8.1. The TIL value is 30 hops. The packet contains 38 bytes: 20 bytes of IP header, 8 bytes of UDP header, and 10 bytes of application data. The application data are used by *traceroute* to keep track of the packets.

The first line shows the first router visited. The router is named Dcore.fhda.edu with IP address 153.18.31.254. The first round-trip time was 0.995 ms, the second was 0.899 ms, and the third was 0.878 ms.

The second line shows the second router visited. The router is named Dbackup.fhda.edu with IP address 153.18.251.4. The three round-trip times are also shown.

The third line shows the destination host. We know that this is the destination host because there are no more lines. The destination host is the server thda.edu, but it is named tiptoe.fhda.edu with the IP address 153.18.8.1. The three round-trip times are also shown.

Example 21.5

In this example, we trace a longer route, the route to xerox.com.

```
$ traceroute xerox.com
traceroute to xerox.com (13.1.64.93), 30 hops max, 38 byte packets
 1  Dcore.fbda.edu      (153.18.31.254)    0.622 ms   0.891 ms   0.875 ms
 2  Ddmz.fbda.edu       (153.18.251.40)    2.132 ms   2.266 ms   2.094ms
 3  Cinic.fhda.edu      (153.18.253.126)   2.110 ms   2.145 ms   1.763 ms
 4  cenic.net           (137.164.32.140)   3.069 ms   2.875 ms   2.930ms
 5  cenic.net           (137.164.22.31)    4.205 ms   4.870 ms   4.197 ms

14  snfc21.pbi.net      (151.164.191.49)   7.656 ms   7.129 ms   6.866ms
15  sbcglobaLnet        (151.164.243.58)   7.844 ms   7.545 ms   7.353 ms
16  pacbell.net         (209.232.138.114)  9.857 ms   9.535 ms   9.603 ms
17  209.233.48.223      (209.233.48.223)   10.634ms   10.771 ms  10.592 ms
18  alpha.Xerox.COM     (13.1.64.93)       11.172 ms  11.048 ms  10.922ms
```

Here there are 17 hops between source and destination. Note that some round-trip times look unusual. It could be that a router was too busy to process the packet immediately.

21.3 IGMP

The IP protocol can be involved in two types of communication: unicasting and multicasting. Unicasting is the communication between one sender and one receiver. It is a one-to-one communication. However, some processes sometimes need to send the same message to a large number of receivers simultaneously. This is called multicasting, which is a one-to-many communication. Multicasting has many applications. For example, multiple stockbrokers can simultaneously be informed of changes in a stock price, or travel agents can be informed of a plane cancellation. Some other applications include distance learning and video-on-demand.

The Internet Group Management Protocol (IGMP) is one of the necessary, but not sufficient (as we will see), protocols that is involved in multicasting. IGMP is a companion to the IP protocol.

Group Management

For multicasting in the Internet we need routers that are able to route multicast packets. The routing tables of these routers must be updated by using one of the multicasting routing protocols that we discuss in Chapter 22.

IGMP is not a multicasting routing protocol; it is a protocol that manages group membership. In any network, there are one or more multicast routers that distribute multicast packets to hosts or other routers. The IGMP protocol gives the multicast routers information about the membership status of hosts (routers) connected to the network.

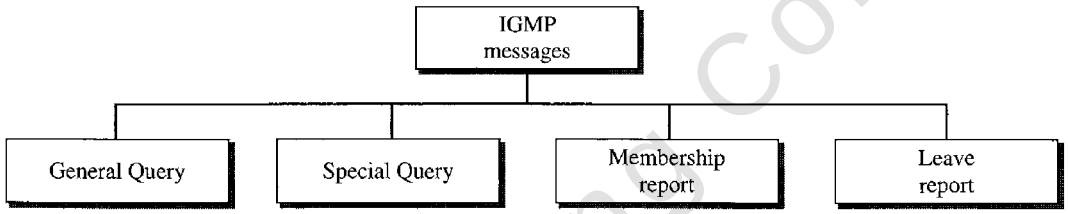
A multicast router may receive thousands of multicast packets every day for different groups. If a router has no knowledge about the membership status of the hosts, it must broadcast all these packets. This creates a lot of traffic and consumes bandwidth. A better solution is to keep a list of groups in the network for which there is at least one loyal member. IGMP helps the multicast router create and update this list.

IGMP is a group management protocol. It helps a multicast router create and update a list of loyal members related to each router interface.

IGMP Messages

IOMP has gone through two versions. We discuss IOMPv2, the current version. IOMPv2 has three types of messages: the query, the membership report, and the leave report. There are two types of query messages: general and special (see Figure 21.16).

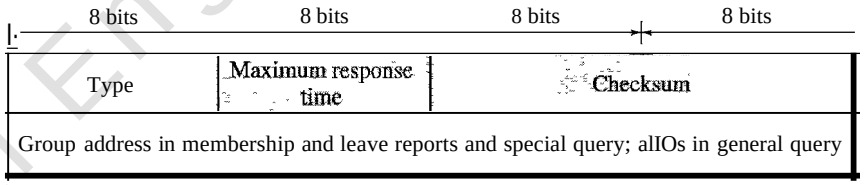
Figure 21.16 *IGMP message types*



Message Format

Figure 21.17 shows the format of an IOMP (version 2) message.

Figure 21.17 *IGMP message fonnat*



- Type. This 8-bit field defines the type of message, as shown in Table 21.1. The value of the type is shown in both hexadecimal and binary notation.

Table 21.1 *IGMP type field*

Type	*Value*
General or special query	0x11 or 00010001
Membership report	0x16 or 00010110
Leave report	0x17 or 00010111

- Maximum Response Time. This 8-bit field defines the amount of time in which a query must be answered. The value is in tenths of a second; for example, if the

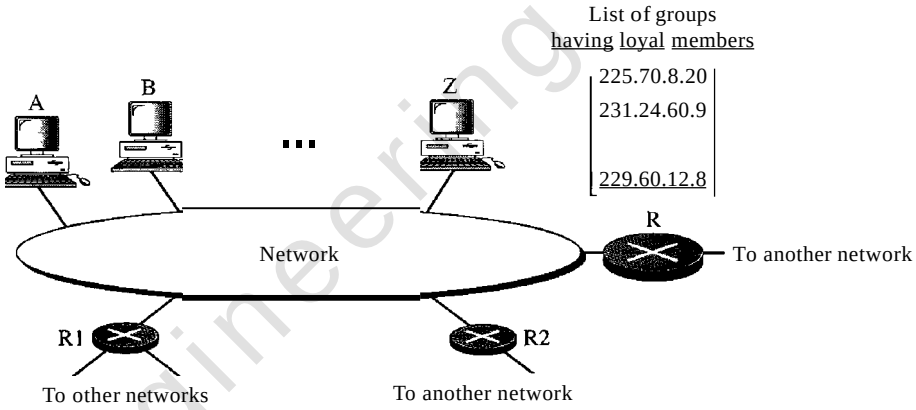
value is 100, it means 10 s. The value is nonzero in the query message; it is set to zero in the other two message types. We will see its use shortly.

- Checksum. This is a 16-bit field carrying the checksum. The checksum is calculated over the 8-byte message.
- Group address. The value of this field is 0 for a general query message. The value defines the groupid (multicast address of the group) in the special query, the membership report, and the leave report messages.

IGMP Operation

IGMP operates locally. A multicast router connected to a network has a list of multicast addresses of the groups with at least one loyal member in that network (see Figure 21.18).

Figure 21.18 *IGMP operation*



For each group, there is one router that has the duty of distributing the multicast packets destined for that group. This means that if there are three multicast routers connected to a network, their lists of groupids are mutually exclusive. For example, in Figure 21.18 only router R distributes packets with the multicast address of225.70.8.20.

A host or multicast router can have membership in a group. When a host has membership, it means that one of its processes (an application program) receives multicast packets from some group. When a router has membership, it means that a network connected to one of its other interfaces receives these multicast packets. We say that the host or the router has an *interest* in the group. In both cases, the host and the router keep a list of groupids and relay their interest to the distributing router.

For example, in Figure 21.18, router R is the distributing router. There are two other multicast routers (R1 and R2) that, depending on the group list maintained by router R, could be the recipients of router R in this network. Routers R1 and R2 may be distributors for some of these groups in other networks, but not on this network.

Joining a Group

A host or a router can join a group. A host maintains a list of processes that have membership in a group. When a process wants to join a new group, it sends its request to the host.

The host adds the name of the process and the name of the requested group to its list. If this is the first entry for this particular group, the host sends a membership report message. If this is not the first entry, there is no need to send the membership report since the host is already a member of the group; it already receives multicast packets for this group.

The protocol requires that the membership report be sent twice, one after the other within a few moments. In this way, if the first one is lost or damaged, the second one replaces it.

In IGMP, a membership report is sent twice, one after the other.

Leaving a Group

When a host sees that no process is interested in a specific group, it sends a leave report. Similarly, when a router sees that none of the networks connected to its interfaces is interested in a specific group, it sends a leave report about that group.

However, when a multicast router receives a leave report, it cannot immediately purge that group from its list because the report comes from just one host or router; there may be other hosts or routers that are still interested in that group. To make sure, the router sends a special query message and inserts the groupid, or multicast address, related to the group. The router allows a specified time for any host or router to respond. If, during this time, no interest (membership report) is received, the router assumes that there are no loyal members in the network and purges the group from its list.

Monitoring Membership

A host or router can join a group by sending a membership report message. It can leave a group by sending a leave report message. However, sending these two types of reports is not enough. Consider the situation in which there is only one host interested in a group, but the host is shut down or removed from the system. The multicast router will never receive a leave report. How is this handled? The multicast router is responsible for monitoring all the hosts or routers in a LAN to see if they want to continue their membership in a group.

The router periodically (by default, every 125 s) sends a general query message. In this message, the group address field is set to 0.0.0.0. This means the query for membership continuation is for all groups in which a host is involved, not just one.

The general query message does not define a particular group.

The router expects an answer for each group in its group list; even new groups may respond. The query message has a maximum response time of 10 s (the value of the field is actually 100, but this is in tenths of a second). When a host or router receives the general query message, it responds with a membership report if it is interested in a group. However, if there is a common interest (two hosts, for example, are interested in the same group), only one response is sent for that group to prevent unnecessary traffic. This is called a delayed response. Note that the query message must be sent by only one

router (normally called the query router), also to prevent unnecessary traffic. We discuss this issue shortly.

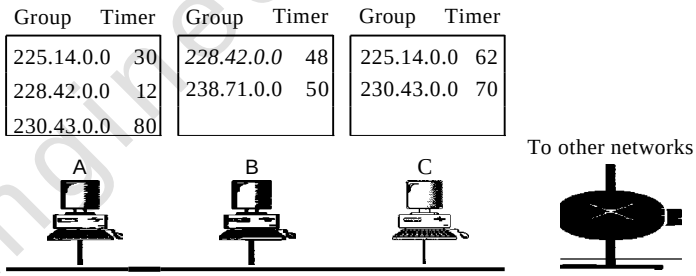
Delayed Response

To prevent unnecessary traffic, IGMP uses a delayed response strategy. When a host or router receives a query message, it does not respond immediately; it delays the response. Each host or router uses a random number to create a timer, which expires between 1 and 10s. The expiration time can be in steps of 1 s or less. A timer is set for each group in the list. For example, the timer for the first group may expire in 2 s, but the timer for the third group may expire in 5 s. Each host or router waits until its timer has expired before sending a membership report message. During this waiting time, if the timer of another host or router, for the same group, expires earlier, that host or router sends a membership report. Because, as we will see shortly, the report is broadcast, the waiting host or router receives the report and knows that there is no need to send a duplicate report for this group; thus, the waiting station cancels its corresponding timer.

Example 21.6

Imagine there are three hosts in a network, as shown in Figure 21.19.

Figure 21.19 *Example 21.6*



A query message was received at time 0; the random delay time (in tenths of seconds) for each group is shown next to the group address. Show the sequence of report messages.

Solution

The events occur in this sequence:

a. Time 12: The timer for 228.42.0.0 in host A expires, and a membership report is sent, which is received by the router and every host including host B which cancels its timer for 228.42.0.0.

b. Time 30: The timer for 225.14.0.0 in host A expires, and a membership report is sent, which is received by the router and every host including host C which cancels its timer for 225.14.0.0.

c. Time 50: The timer for 238.71.0.0 in host B expires, and a membership report is sent, which is received by the router and every host.

d. Time 70: The timer for 230.43.0.0 in host C expires, and a membership report is sent, which is received by the router and every host including host A which cancels its timer for 230.43.0.0.

Note that if each host had sent a report for every group in its list, there would have been seven reports; with this strategy only four reports are sent.

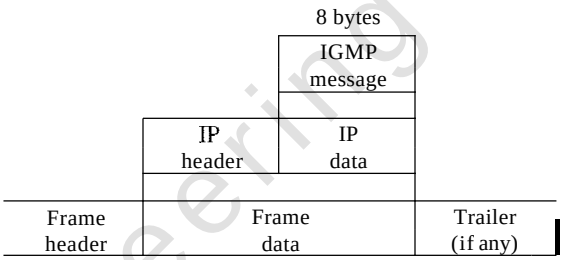
Query Router

Query messages may create a lot of responses. To prevent unnecessary traffic, IGMP designates one router as the query router for each network. Only this designated router sends the query message, and the other routers are passive (they receive responses and update their lists).

Encapsulation

The IGMP message is encapsulated in an IP datagram, which is itself encapsulated in a frame. See Figure 21.20.

Figure 21.20 *Encapsulation of IGMP packet*



Encapsulation at Network Layer

The value of the protocol field is 2 for the IGMP protocol. Every IP packet carrying this value in its protocol field has data delivered to the IGMP protocol. When the message is encapsulated in the IP datagram, the value of TTL must be 1. This is required because the domain of IGMP is the LAN. No IGMP message must travel beyond the LAN. A TTL value of 1 guarantees that the message does not leave the LAN since this value is decremented to 0 by the next router and, consequently, the packet is discarded. Table 21.2 shows the destination IP address for each type of message.

The IP packet that carries an IGMP packet has a value of 1 in its TTL field.

Table 21.2 *Destination IP addresses*

Type	*IP Destination Address*
Query	224.0.0.1 All systems on this subnet
Membership report	The multicast address of the group
Leave report	224.0.0.2 All routers on this subnet

A query message is multicast by using the multicast address 224.0.0.1 All hosts and all routers will receive the message. A membership report is multicast using a

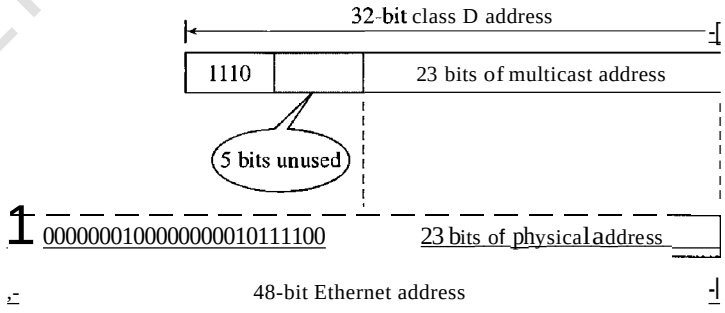
destination address equal to the multicast address being reported (groupid). Every station (host or router) that receives the packet can immediately determine (from the header) the group for which a report has been sent. As discussed previously, the timers for the corresponding unsent reports can then be canceled. Stations do not need to open the packet to find the groupid. This address is duplicated in a packet; it's part of the message itself and also a field in the IP header. The duplication prevents errors. A leave report message is multicast using the multicast address 224.0.0.2 (all routers on this subnet) so that routers receive this type of message. Hosts receive this message too, but disregard it.

Encapsulation at Data Link Layer

At the network layer, the IGMP message is encapsulated in an IP packet and is treated as an IP packet. However, because the IP packet has a multicast IP address, the ARP protocol cannot find the corresponding MAC (physical) address to forward the packet at the data link layer. What happens next depends on whether the underlying data link layer supports physical multicast addresses.

Physical Multicast Support Most LANs support physical multicast addressing. Ethernet is one of them. An Ethernet physical address (MAC address) is six octets (48 bits) long. If the first 25 bits in an Ethernet address are 00000001000000000010111100, this identifies a physical multicast address for the TCP/IP protocol. The remaining 23 bits can be used to define a group. To convert an IP multicast address into an Ethernet address, the multicast router extracts the least significant 23 bits of a class D IP address and inserts them into a multicast Ethernet physical address (see Figure 21.21).

Figure 21.21 *Mapping class D to Ethernet physical address*



However, the group identifier of a class D IP address is 28 bits long, which implies that 5 bits is not used. This means that 32 (25) multicast addresses at the IP level are mapped to a single multicast address. In other words, the mapping is many-to-one instead of one-to-one. If the 5 leftmost bits of the group identifier of a class D address are not all zeros, a host may receive packets that do not really belong to the group in which it is involved. For this reason, the host must check the IP address and discard any packets that do not belong to it.

Other LANs support the same concept but have different methods of mapping.

An Ethernet multicast physical address is in the range
01:00:5E:00:OO:OO to 01:00:5E:7F:FF:FF.

Example 21.7

Change the multicast IP address 230.43.14.7 to an Ethernet multicast physical address.

Solution

We can do this in two steps:

a. We write the rightmost 23 bits of the IP address in hexadecimal. This can be done by changing the rightmost 3 bytes to hexadecimal and then subtracting 8 from the leftmost digit if it is greater than or equal to 8. In our example, the result is 2B:OE:07.

b. We add the result of part a to the starting Ethernet multicast address, which is 01:00:5E:00:00:00. The result is

01:00:5E:2B:OE:07

Example 21.8

Change the multicast IP address 238.212.24.9 to an Ethernet multicast address.

Solution

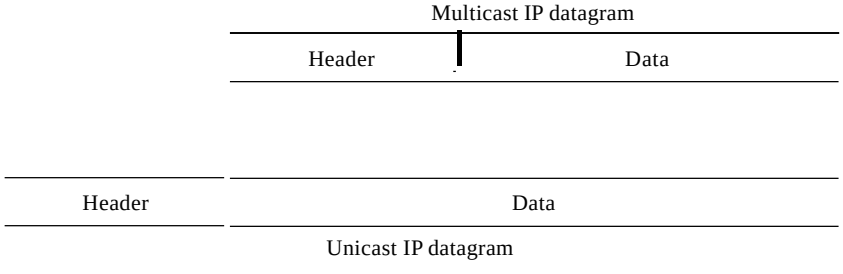
a. The rightmost 3 bytes in hexadecimal is D4: 18:09. We need to subtract 8 from the leftmost digit, resulting in 54:18:09.

b. We add the result of part a to the Ethernet multicast starting address. The result is

01:00:5E:54: 18:09

No Physical Multicast Support Most WANs do not support physical multicast addressing. To send a multicast packet through these networks, a process called *tunneling* is used. In tunneling, the multicast packet is encapsulated in a unicast packet and sent through the network, where it emerges from the other side as a multicast packet (see Figure 21.22).

Figure 21.22 *Tunneling*



Netstat **Utility**

The *netstat* utility can be used to find the multicast addresses supported by an interface.

Example 21.9

We use *netstat* with three options: -n, -r, and -a. The -n option gives the numeric versions of IP addresses, the -r option gives the routing table, and the -a option gives all addresses (unicast and multicast). Note that we show only the fields relative to our discussion. "Gateway" defines the router, "Iface" defines the interface.

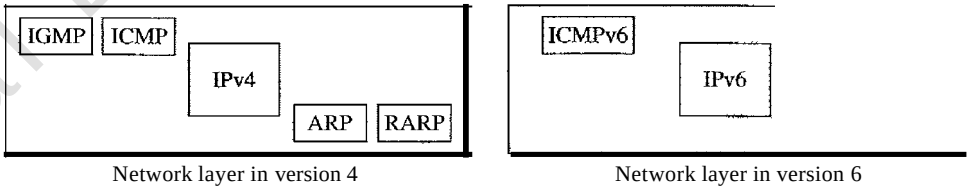
```
$ netstat -nra
Kernel IP routing table
Destination     Gateway         Mask            Flags   Iface
153.18.16.0     0.0.0.0         255.255.240.0   U       ethO
169.254.0.0     0.0.0.0         255.255.0.0     U       ethO
127.0.0.0       0.0.0.0         255.0.0.0       U       10
224.0.0.0       0.0.0.0         224.0.0.0       u       ethO
0.0.0.0         153.18.31.254   0.0.0.0         va      ethO
```

Nate that the multicast address is shown in color. Any packet with a multicast address from 224.0.0.0 to 239.255.255.255 is masked and delivered to the Ethernet interface.

21.4 ICMPv6

We discussed 1Pv6 in Chapter 20. Another protocol that has been modified in version 6 of the TCPIIP protocol suite is ICMP (ICMPv6). This new version follows the same strategy and purposes of version 4. ICMPv4 has been modified to make it more suitable for IPv6. In addition, some protocols that were independent in version 4 are now part of Internetworking Control Message Protocol (ICMPv6). Figure 21.23 compares the network layer of version 4 to version 6.

Figure 21.23 *Comparison of network layers in version 4 and version 6*



The ARP and IGMP protocols in version 4 are combined in ICMPv6. The RARP protocol is dropped from the suite because it was rarely used and BOOTP has the same functionality.

Just as in ICMPv4, we divide the ICMP messages into two categories. However, each category has more types of messages than before.

Error Reporting

As we saw in our discussion of version 4, one of the main responsibilities of ICMP is to report errors. Five types of errors are handled: destination unreachable, packet too big, time exceeded, parameter problems, and redirection. ICMPv6 forms an error packet,

which is then encapsulated in an IP datagram. This is delivered to the original source of the failed datagram. Table 21.3 compares the error-reporting messages of ICMPv4 with ICMPv6. The source-quench message is eliminated in version 6 because the priority and the flow label fields allow the router to control congestion and discard the least important messages. In this version, there is no need to inform the sender to slow down. The packet-too-big message is added because fragmentation is the responsibility of the sender in IPv6. If the sender does not make the right packet size decision, the router has no choice but to drop the packet and send an error message to the sender.

Table 21.3 *Comparison of error-reporting messages in ICMPv4 and ICMPv6*

Type of Message	*Version 4*	*Version 6*
Destination unreachable	Yes	Yes
Source quench	Yes	No
Packet too big	No	Yes
Time exceeded	Yes	Yes
Parameter problem	Yes	Yes
Redirection	Yes	Yes

Destination Unreachable

The concept of the destination-unreachable message is exactly the same as described for ICMP version 4.

Packet Too Big

This is a new type of message added to version 6. If a router receives a datagram that is larger than the maximum transmission unit (MTU) size of the network through which the datagram should pass, two things happen. First, the router discards the datagram and then an ICMP error packet-a packet-too-big message-is sent to the source.

Time Exceeded

This message is similar to the one in version 4.

Parameter Problem

This message is similar to its version 4 counterpart.

Redirection

The purpose of the redirection message is the same as described for version 4.

Query

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages. Four different groups of messages have been defined: echo request and reply, router solicitation and advertisement, neighbor solicitation and advertisement, and group membership. Table 21.4 shows a comparison between

the query messages in versions 4 and 6. Two sets of query messages are eliminated from ICMPv6: time-stamp request and reply- and address-mask request and reply. The time-stamp request and reply messages are eliminated because they are implemented in other protocols such as TCP and because they were rarely used in the past. The address-mask request and reply messages are eliminated in IPv6 because the subnet section of an address allows the subscriber to use up to $2^{32} - 1$ subnets. Therefore, subnet masking, as defined in IPv4, is not needed here.

Table 21.4 *Comparison of query messages in ICMPv4 and ICMPv6*

Type of Message	*Version 4*	*Version 6*
Echo request and reply	Yes	Yes
Timestamp request and reply	Yes	No
Address-mask request and reply	Yes	No
Router solicitation and advertisement	Yes	Yes
Neighbor solicitation and advertisement	ARP	Yes
Group membership	IGMP	Yes

Echo Request and Reply

The idea and format of the echo request and reply messages are the same as those in version 4.

Router Solicitation and Advertisement

The idea behind the router-solicitation and -advertisement messages is the same as in version 4.

Neighbor Solicitation and Advertisement

As previously mentioned, the network layer in version 4 contains an independent protocol called Address Resolution Protocol (ARP). In version 6, this protocol is eliminated, and its duties are included in ICMPv6. The idea is exactly the same, but the format of the message has changed.

Group Membership

As previously mentioned, the network layer in version 4 contains an independent protocol called IGMP. In version 6, this protocol is eliminated, and its duties are included in ICMPv6. The purpose is exactly the same.

21.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and site. The items in brackets [...] refer to the reference list at the end of the text.

Books

ARP and RARP are discussed in Chapter 7 of [For06] and Chapters 4 and 5 of [Ste94]. ICMP is discussed in Chapter 9 of [For06] and Chapter 6 of [Ste94]. IGMP is discussed in Chapter 10 of [For06] and Chapter 13 of [Ste94]. BOOTP and DHCP are discussed in Chapter 16 of [For06] and Chapter 16 of [Ste94]. ICMPv6 is discussed in Chapter 27 of [For06].

Site

- www.ietf.org/rfc.html Information about RFCs

RFCs

A discussion of ARP and RARP can be found in following RFCs:

826,903,925,1027,1293,1329,1433,1868,1931,2390

A discussion of ICMP can be found in the following RFCs:

777,792,1016,1018,1256,1788,2521

A discussion of IGMP can be found in the following RFCs:

966,988,1054,1112, 1301,1458, 1469,1768,2236,2357,2365,2502,2588

A discussion of BOOTP and DHCP can be found in the following RFCs:

951,1048,1084,1395,1497,1531,1532,1533,1534, 1541, 1542,2131,2132

21.6 KEY TERMS

address-mask-reply message
address-mask-request message
address resolution protocol (ARP)
Bootstrap Protocol (BOOTP)
delayed response strategy
destination-unreachable message
dynamic configuration protocol
Dynamic Host Configuration Protocol (DHCP)
dynamic mapping
echo-request and echo-reply messages
error-reporting message
general query message
groupid
group membership
Internet Control Message Protocol (ICMP)
Internet Group Management Protocol (IGMP)
Internetworking Control Message Protocol, version 6 (ICMPv6)
lease
leave report
membership report

multicast address
multicast router
multicasting
neighbor-solicitation and advertisement message
packet-too-big message
parameter-problem message
physical address
proxyARP
query message
query router
redirection message
relay agent
reverse address resolution protocol (RARP)
router-solicitation and router-advertisement messages
source-quench message
special query message
static mapping
time-exceeded message
timestamp request and timestamp reply messages
traceroute
tunneling

21.7 SUMMARY

- Delivery of a packet to a host or router requires two levels of addresses: logical and physical. A physical address identifies a host or router at the physical level.
- Mapping of a logical address to a physical address can be static or dynamic.
- Static mapping involves a list of logical and physical address correspondences; maintenance of the list requires high overhead.
- The address resolution protocol (ARP) is a dynamic mapping method that finds a physical address, given a logical address.
- In proxy ARP, a router represents a set of hosts. When an ARP request seeks the physical address of any host in this set, the router sends its own physical address. This creates a subnetting effect.
- Reverse Address Resolution Protocol (RARP) is a form of dynamic mapping in which a given physical address is associated with a logical address.
- ICMP sends four pairs of query messages: echo-request and echo-reply, time-stamp request and reply, address-mask-request and -reply, and router solicitation and advertisement.
- The checksum for ICMP is calculated by using both the header and the data fields of the ICMP message.
- Packet InterNet Groper *(ping)* is an application program that uses the services of ICMP to test the reachability of a host.
- Multicasting is the sending of the same message to more than one receiver simultaneously.
- The Internet Group Management Protocol (IGMP) helps multicast routers create and update a list of loyal members related to a router interface.
- The three IGMP message types are the query message, the membership report, and the leave report.

- A delayed response strategy prevents unnecessary traffic on a LAN.
- The IGMP message is encapsulated in an IP datagram.
- Most LANs, including Ethernet, support physical multicast addressing.
- WANs that do not support physical multicast addressing can use a process called tunneling to send multicast packets.
- BOOTP and Dynamic Host Configuration Protocol (DHCP) are client/server applications that deliver vital network information to either diskless computers or computers at first boot.
- A BOOTP request is encapsulated in a UDP user datagram.
- BOOTP, a static configuration protocol, uses a table that maps IP addresses to physical addresses.
- A relay agent is a router that helps send local BOOTP requests to remote servers.
- DHCP is a dynamic configuration protocol with two databases: One is similar to BOOTP, and the other is a pool of IP addresses available for temporary assignment.
- The DHCP server issues a lease for an IP address to a client for a specific time.
- ICMPv6, like version 4, reports errors, handles group memberships, updates specific router and host tables, and checks the viability of a host.

21.8 PRACTICE SET

Review Questions

1. Is the size of the ARP packet fixed? Explain.
2. What is the size of an ARP packet when the protocol is IPv4 and the hardware is Ethernet?
3. What is the size of an Ethernet frame carrying an ARP packet in Question 2?
4. What is the broadcast address for Ethernet?
5. Why is there a restriction on the generation of an ICMPv4 message in response to a failed ICMPv4 error message?
6. What is the purpose of including the IPv4 header and the first 8 bytes of datagram data in the error-reporting ICMPv4 messages?
7. Give an example of a situation in which a host would never receive a redirection message.
8. What is the minimum size of an ICMPv4 packet? What is the maximum size of an ICMPv4 packet?
9. What is the minimum size of an IPv4 packet that carries an ICMPv4 packet? What is the maximum size?
10. How can we determine if an IPv4 packet is carrying an ICMPv4 packet?
11. What is the minimum size of an Ethernet frame that carries an IPv4 packet which in turn carries an ICMPv4 packet? What is the maximum size?
12. Why is there no need for the ICMPv4 message to travel outside its own network?

Exercises

13. A router with IPv4 address 125.45.23.12 and Ethernet physical address 23:45:AB:4F:67:CD has received a packet for a host destination with IP address 125.11.78.10. Show the entries in the ARP request packet sent by the router. Assume no subnetting.
14. Show the entries in the ARP packet sent in response to Exercise 13.
15. Encapsulate the result of Exercise 13 in a data link frame. Fill in all the fields.
16. Encapsulate the result of Exercise 14 in a data link frame. Fill in all the fields.
17. Host A sends a datagram to host B. Host B never receives the datagram, and host A never receives notification of failure. Give two different explanations of what might have happened.
18. Calculate the checksum for the following ICMP packet:
    Type: Echo Request Identifier: 123 Sequence number: 25 Message: Hello
19. A router receives an IPv4 packet with source IP address 130.45.3.3 and destination IP address 201.23.4.6. The router cannot find the destination IP address in its routing table. Which ICMPv4 message should be sent?
20. TCP receives a segment with destination port address 234. TCP checks and cannot find an open port for this destination. Which ICMPv4 message should be sent?
21. A multicast address for a group is 231.24.60.9. What is its 48-bit Ethernet address for a LAN using TCPIIP?
22. If a router has 20 entries in its group table, should it send 20 different queries periodically or just one? Explain your answer.
23. If a host wants to continue membership in five groups, should it send five different membership report messages or just one?
24. A router on an Ethernet network has received a multicast IP packet with groupid 226.17.18.4. When the host checks its multicast group table, it finds this address. Show how the router sends this packet to the recipients by encapsulating the IP packet in an Ethernet frame. Show all the entries of the Ethernet frame. The outgoing IP address of the router is 185.23.5.6, and its outgoing physical address is 4A224512E1E2. Does the router need the services of ARP?
25. A host with IPv4 address 114.45.7.9 receives an IGMP query. When it checks its group table, it finds no entries. What action should the host take? Should it send any messages?
26. A host with IPv4 address 222.5.7.19 receives an IGMP query. When it checks its routing table, it finds two entries in its table: 227.4.3.7 and 229.45.6.23. What action should the host take? Should it send any messages? If so, what type and how many?
27. How many multicast addresses can be supported for the IPv4 protocol in Ethernet? How many multicast addresses can be supported by the IPv4 protocol? What is the size of address space lost when we transform a multicast IPv4 address to an Ethernet multicast address?
28. Change the following IPv4 multicast addresses to Ethernet multicast addresses. How many of them specify the same Ethernet address?

a. 224.18.72.8
b. 235.18.72.8
c. 237.18.6.88
d. 224.88.12.8

Research Activities

29. Use the *ping* program to test your own computer (loopback).
30. Use the *ping* program to test a host inside the United States.
31. Use the *ping* program to test a host outside the United States.
32. Use *traceroute* (or *tracert)* to find the route from your computer to a computer in a college or university.
33. Use *netstat* to find out if your server supports multicast addressing.
34. DHCP uses several messages such as DHCPREQUEST, DHCPDECLINE, DHCPACK, DHCPNACK, and DHCPRELEASE. Find the purpose of these messages.

CHAPTER 22

Network Layer: Delivery, Forwarding, and Routing

This chapter describes the delivery, forwarding, and routing of IP packets to their final destinations. Delivery refers to the way a packet is handled by the underlying networks under the control of the network layer. Forwarding refers to the way a packet is delivered to the next station. Routing refers to the way routing tables are created to help in forwarding.

Routing protocols are used to continuously update the routing tables that are consulted for forwarding and routing. In this chapter, we also briefly discuss common unicast and multicast routing protocols.

22.1 DELIVERY

The network layer supervises the handling of the packets by the underlying physical networks.We define this handling as the delivery of a packet.

Direct Versus Indirect Delivery

The delivery of a packet to its final destination is accomplished by using two different methods of delivery, direct and indirect, as shown in Figure 22.1.

Direct Delivery

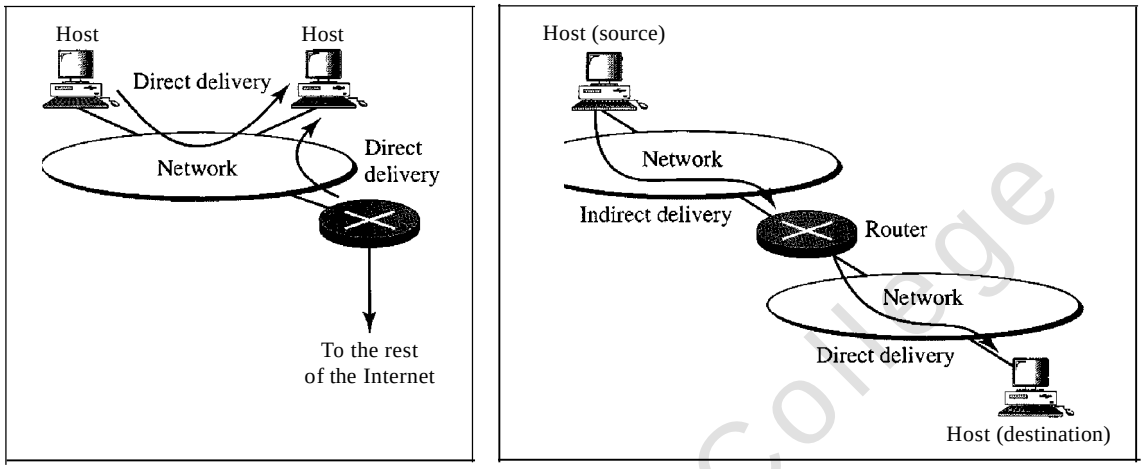
In a direct delivery, the final destination of the packet is a host connected to the same physical network as the deliverer. Direct delivery occurs when the source and destination of the packet are located on the same physical network or when the delivery is between the last router and the destination host.

The sender can easily determine if the delivery is direct. It can extract the network address of the destination (using the mask) and compare this address with the addresses of the networks to which it is connected. If a match is found, the delivery is direct.

Indirect Delivery

If the destination host is not on the same network as the deliverer, the packet is delivered indirectly. In an indirect delivery, the packet goes from router to router until it reaches the one connected to the same physical network as its final destination. Note

Figure 22.1 *Direct and indirect delivery*



a. Direct delivery

b. Indirect and direct delivery

that a delivery always involves one direct delivery but zero or more indirect deliveries. Note also that the last delivery is always a direct delivery.

22.2 FORWARDING

Forwarding means to place the packet in its route to its destination. Forwarding requires a host or a router to have a routing table. When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the route to the final destination. However, this simple solution is impossible today in an internetwork such as the Internet because the number of entries needed in the routing table would make table lookups inefficient.

Forwarding Techniques

Several techniques can make the size of the routing table manageable and also handle issues such as security. We briefly discuss these methods here.

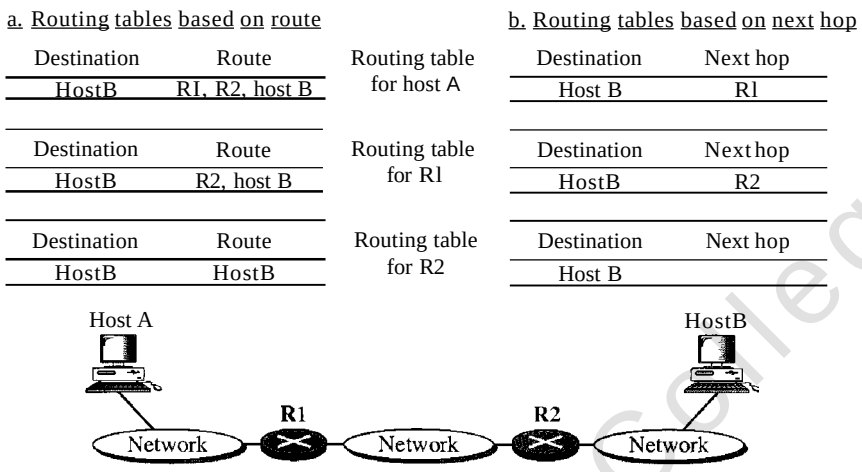
Next-Hop Method Versus Route Method

One technique to reduce the contents of a routing table is called the next-hop method. In this technique, the routing table holds only the address of the next hop instead of information about the complete route (route method). The entries of a routing table must be consistent with one another. Figure 22.2 shows how routing tables can be simplified by using this technique.

Network-Specific Method Versus Host-Specific Method

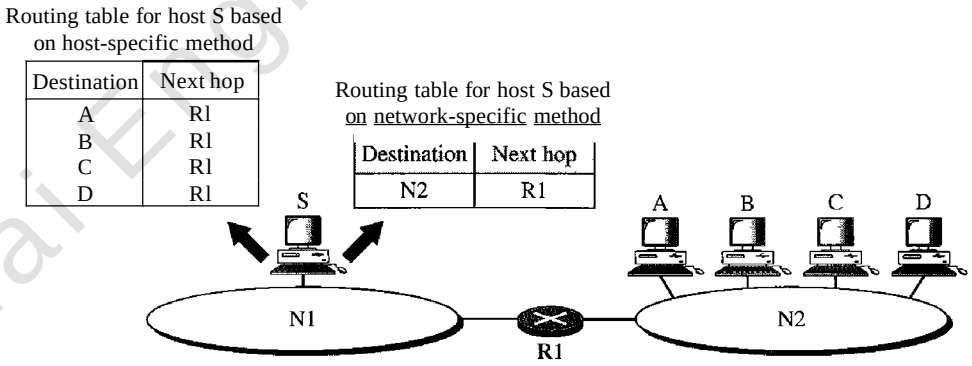
A second technique to reduce the routing table and simplify the searching process is called the network-specific method. Here, instead of having an entry for every destination host connected to the same physical network (host-specific method), we have

Figure 22.2 *Route method versus next-hop method*



only one entry that defines the address of the destination network itself. In other words, we treat all hosts connected to the same network as one single entity. For example, if 1000 hosts are attached to the same network, only one entry exists in the routing table instead of 1000. Figure 22.3 shows the concept.

Figure 22.3 *Host-specific versus network-specific method*

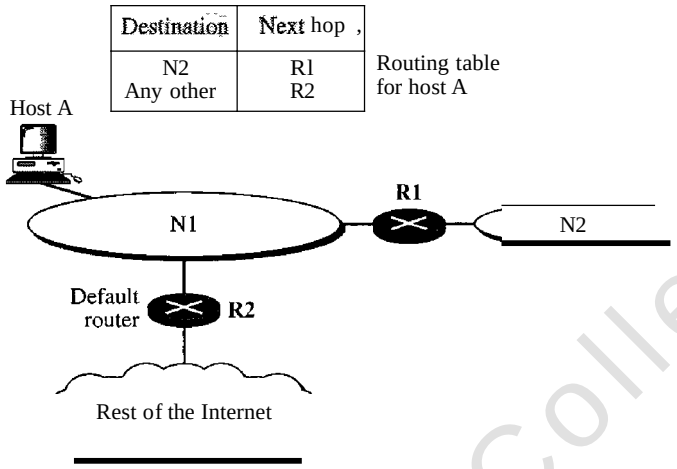


Host-specific routing is used for purposes such as checking the route or providing security measures.

Default Method

Another technique to simplify routing is called the default method. In Figure 22.4 host A is connected to a network with two routers. Router R1 routes the packets to hosts connected to network N2. However, for the rest of the Internet, router R2 is used. So instead of listing all networks in the entire Internet, host A can just have one entry called the *default* (normally defined as network address 0.0.0.0).

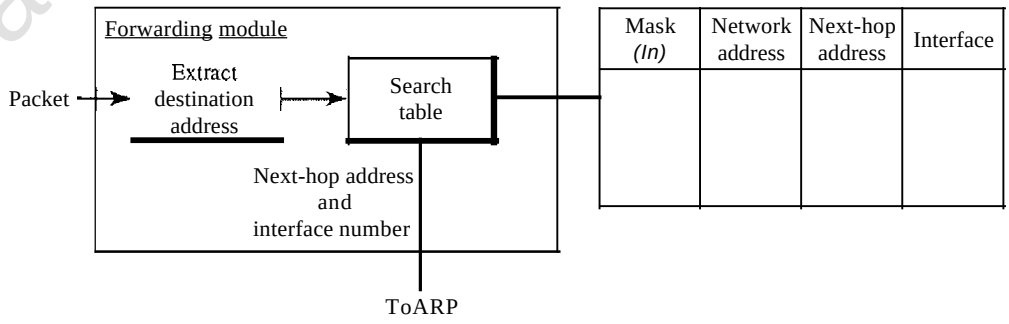
Figure 22.4 *Default method*



Forwarding Process

Let us discuss the forwarding process. We assume that hosts and routers use classless addressing because classful addressing can be treated as a special case of classless addressing. In classless addressing, the routing table needs to have one row of information for each block involved. The table needs to be searched based on the network address (first address in the block). Unfortunately, the destination address in the packet gives no clue about the network address. To solve the problem, we need to include the mask *(In)* in the table; we need to have an extra column that includes the mask for the corresponding block. Figure 22.5 shows a simple forwarding module for classless addressing.

Figure 22.5 *Simplified forwarding module in classless address*



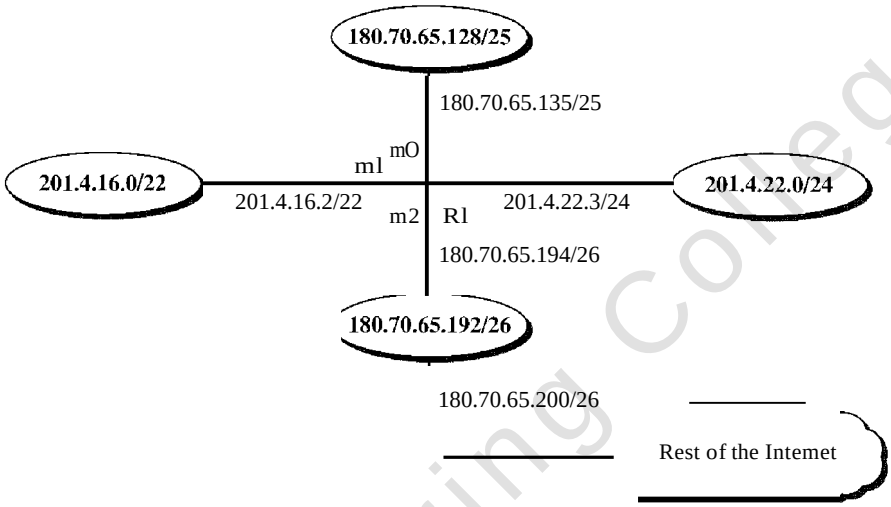
Note that we need at least four columns in our routing table; usually there are more.

In classless addressing, we need at least four colwnns in a routing table.

Example 22.1

Make a routing table for router Rl, using the configuration in Figure 22.6.

Figure 22.6 *Configurationfor Example 22.1*



Solution

Table 22.1 shows the corresponding table.

Table 22.1 *Routing table for router RI in Figure 22.6*

Mask	*Network Address*	*Next Hop*	*Interface*
/26	180.70.65.192	-	m2
/25	180.70.65.128	-	mO
/24	201.4.22.0	-	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

Example 22.2

Show the forwarding process if a packet arrives at Rl in Figure 22.6 with the destination address 180.70.65.140.

Solution

The router performs the following steps:

1. The first mask (/26) is applied to the destination address. The result is 180.70.65.128, which does not match the corresponding network address.
2. The second mask (/25) is applied to the destination address. The result is 180.70.65.128, which matches the corresponding network address. The next-hop address (the destination address of the packet in this case) and the interface number mO are passed to ARP for further processing.

Example 22.3

Show the forwarding process if a packet arrives at R1 in Figure 22.6 with the destination address 201.4.22.35.

Solution

The router performs the following steps:

1. The first mask (/26) is applied to the destination address. The result is 201.4.22.0, which does not match the corresponding network address (row 1).
2. The second mask (/25) is applied to the destination address. The result is 201.4.22.0, which does not match the corresponding network address (row 2).
3. The third mask (/24) is applied to the destination address. The result is 201.4.22.0, which matches the corresponding network address. The destination address of the packet and the interface number m3 are passed to ARP.

Example 22.4

Show the forwarding process if a packet arrives at R1 in Figure 22.6 with the destination address 18.24.32.78.

Solution

This time all masks are applied, one by one, to the destination address, but no matching network address is found. When it reaches the end of the table, the module gives the next-hop address 180.70.65.200 and interface number m2 to ARP. This is probably an outgoing package that needs to be sent, via the default router, to someplace else in the Internet.

Address Aggregatioll

When we use classless addressing, it is likely that the number of routing table entries will increase. This is so because the intent of classless addressing is to divide up the whole address space into manageable blocks. The increased size of the table results in an increase in the amount of time needed to search the table. To alleviate the problem, the idea of address aggregation was designed. In Figure 22.7 we have two routers.

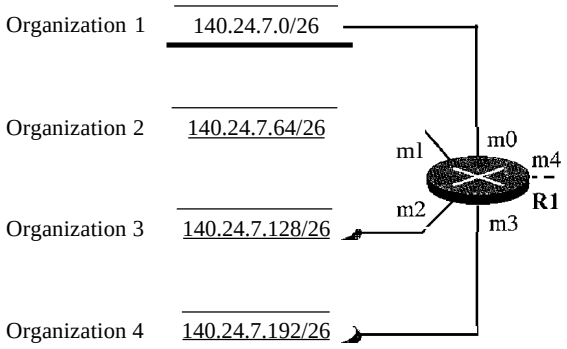
Router R1 is connected to networks of four organizations that each use 64 addresses. Router R2 is somewhere far from R1. Router R1 has a longer routing table because each packet must be correctly routed to the appropriate organization. Router R2, on the other hand, can have a very small routing table. For R2, any packet with destination 140.24.7.0 to 140.24.7.255 is sent out from interface mO regardless of the organization number. This is called address aggregation because the blocks of addresses for four organizations are aggregated into one larger block. Router R2 would have a longer routing table if each organization had addresses that could not be aggregated into one block.

Note that although the idea of address aggregation is similar to the idea of subnetting, we do not have a common site here; the network for each organization is independent. In addition, we can have several levels of aggregation.

Longest Mask Matching

What happens if one of the organizations in Figure 22.7 is not geographically close to the other three? For example, if organization 4 cannot be connected to router R1 for some reason, can we still use the idea of address aggregation and still assign block 140.24.7.192/26 to organization 4?

Figure 22.7 *Address aggregation*



Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	mO
/26	140.24.7.64	----------	ml
/26	140.24.7.128	----------	m2
/26	140.24.7.192	----------	m3
/0	0.0.0.0	Default	m4

Routing table for Rl

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	----------	mO
/0	0.0.0.0	Default	ml

Routmg table for R2

The answer is yes because routing in classless addressing uses another principle, **longest** mask **matching.** This principle states that the routing table is sorted from the longest mask to the shortest mask. In other words, if there are three masks *127, 126,* and *124,* the mask /27 must be the first entry and *124* must be last. Let us see if this principle solves the situation in which organization 4 is separated from the other three organizations. Figure 22.8 shows the situation.

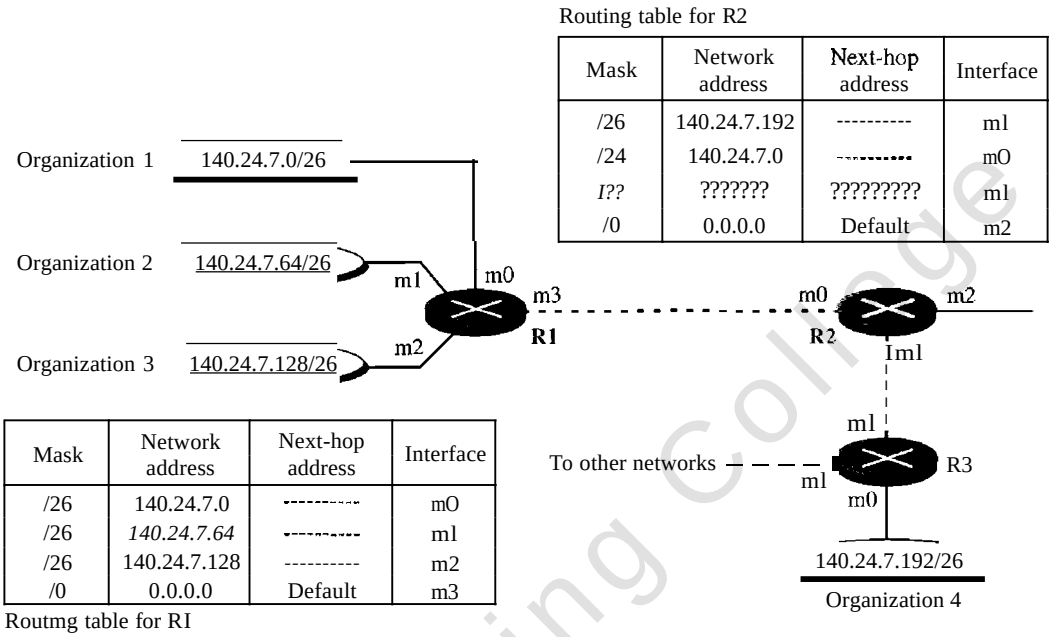
Suppose a packet arrives for organization 4 with destination address 140.24.7.200. The first mask at router R2 is applied, which gives the network address 140.24.7.192. The packet is routed correctly from interface ml and reaches organization 4. If, however, the routing table was not stored with the longest prefix first, applying the /24 mask would result in the incorrect routing of the packet to router Rl.

Hierarchical Routing

To solve the problem of gigantic routing tables, we can create a sense of hierarchy in the routing tables. In Chapter 1, we mentioned that the Internet today has a sense of hierarchy. We said that the Internet is divided into international and national ISPs. National ISPs are divided into regional ISPs, and regional ISPs are divided into local ISPs. If the routing table has a sense of hierarchy like the Internet architecture, the routing table can decrease in size.

Let us take the case of a local ISP. A local ISP can be assigned a single, but large block of addresses with a certain prefix length. The local ISP can divide this block into smaller blocks of different sizes and can assign these to individual users and organizations, both large and small. If the block assigned to the local ISP starts with a.b.c.dln, the ISP can create blocks starting with eJ.g.h/m, where *m* may vary for each customer and is greater than *n.*

Figure 22.8 *Longest mask matching*



Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	ml
/24	140.24.7.0	----------	mO
I??	???????	?????????	ml
/0	0.0.0.0	Default	m2

Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	mO
/26	*140.24.7.64*	----------	ml
/26	140.24.7.128	----------	m2
/0	0.0.0.0	Default	m3

Routmg table for RI

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	mO
I??	???????	?????????	rnl
/0	0.0.0.0	Default	rn2

Routing table for R3

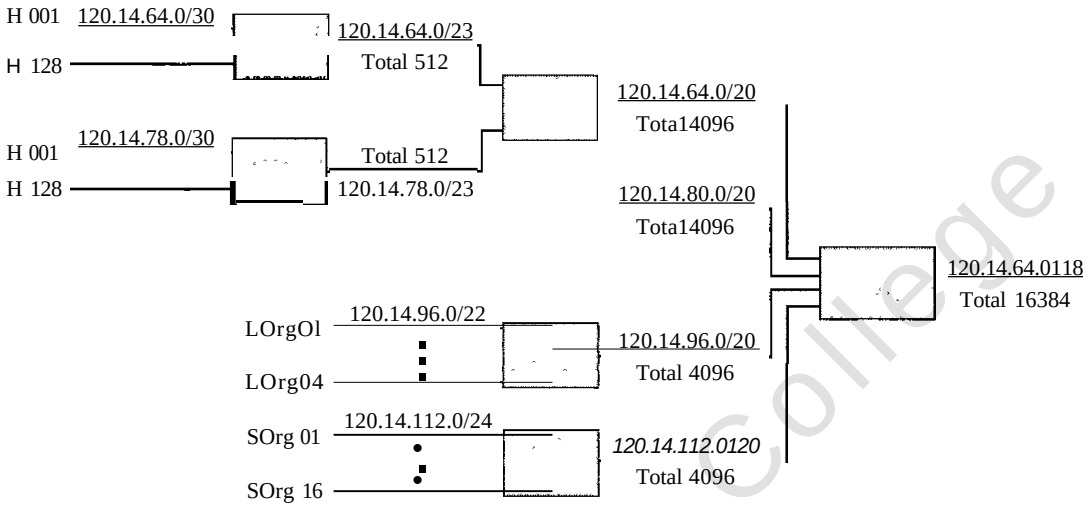
How does this reduce the size of the routing table? The rest of the Internet does not have to be aware of this division. All customers of the local ISP are defined as a.b.c.dln to the rest of the Internet. Every packet destined for one of the addresses in this large block is routed to the local ISP. There is only one entry in every router in the world for all these customers. They all belong to the same group. Of course, inside the local ISp, the router must recognize the subblocks and route the packet to the destined customer. If one of the customers is a large organization, it also can create another level of hierarchy by subnetting and dividing its subblock into smaller subblocks (or sub-subblocks). In classless routing, the levels of hierarchy are unlimited so long as we follow the rules of classless addressing.

Example 22.5

As an example of hierarchical routing, let us consider Figure 22.9. A regional ISP is granted 16,384 addresses starting from 120.14.64.0. The regional ISP has decided to divide this block into four subblocks, each with 4096 addresses. Three of these subblocks are assigned to three local ISPs; the second subblock is reserved for future use. Note that the mask for each block is *120* because the original block with mask /18 is divided into 4 blocks.

The first local ISP has divided its assigned subblock into 8 smaller blocks and assigned each to a small ISP. Each small ISP provides services to 128 households (HOOI to HI28), each using four addresses. Note that the mask for each small ISP is now *123* because the block is further divided into 8 blocks. Each household has a mask of *130,* because a household has only four addresses (2^{32-30} is 4).

Figure 22.9 *Hierarchical routing with ISPs*



The second local ISP has divided its block into 4 blocks and has assigned the addresses to four large organizations (LOrg01 to LOrg04). Note that each large organization has 1024 addresses, and the mask is /22.

The third local ISP has divided its block into 16 blocks and assigned each block to a small organization (SOrg01 to SOrg16). Each small organization has 256 addresses, and the mask is /24.

There is a sense of hierarchy in this configuration. All routers in the Internet send a packet with destination address 120.14.64.0 to 120.14.127.255 to the regional ISP.

The regional ISP sends every packet with destination address 120.14.64.0 to 120.14.79.255 to local ISPl. Local ISP1 sends every packet with destination address 120.14.64.0 to 120.14.64.3 to H001.

Geographical Routing

To decrease the size of the routing table even further, we need to extend hierarchical routing to include geographical routing. We must divide the entire address space into a few large blocks. We assign a block to North America, a block to Europe, a block to Asia, a block to Africa, and so on. The routers of ISPs outside Europe will have only one entry for packets to Europe in their routing tables. The routers of ISPs outside North America will have only one entry for packets to North America in their routing tables. And so on.

Routing Table

Let us now discuss routing tables. A host or a router has a routing table with an entry for each destination, or a combination of destinations, to route IP packets. The routing table can be either static or dynamic.

Static Routing Table

A **static routing table** contains information entered manually. The administrator enters the route for each destination into the table. When a table is created, it cannot update

automatically when there is a change in the Internet. The table must be manually altered by the administrator.

A static routing table can be used in a small internet that does not change very often, or in an experimental internet for troubleshooting. It is poor strategy to use a static routing table in a big internet such as the Internet.

Dynamic Routing Table

A dynamic routing table is updated periodically by using one of the dynamic routing protocols such as RIP, OSPF, or BGP. Whenever there is a change in the Internet, such as a shutdown of a router or breaking of a link, the dynamic routing protocols update all the tables in the routers (and eventually in the host) automatically.

The routers in a big internet such as the Internet need to be updated dynamically for efficient delivery of the IP packets. We discuss in detail the three dynamic routing protocols later in the chapter.

Format

As mentioned previously, a routing table for classless addressing has a minimum of four columns. However, some of today's routers have even more columns. We should be aware that the number of columns is vendor-dependent, and not all columns can be found in all routers. Figure 22.10 shows some common fields in today's routers.

Figure 22.10 *Common fields in a routing table*

Mask	Network address	Next-hop address	Interlace		Reference count	Use

- Mask. This field defines the mask applied for the entry.
- Network address. This field defines the network address to which the packet is finally delivered. In the case of host-specific routing, this field defines the address of the destination host.
- Next-hop address. This field defines the address of the next-hop router to which the packet is delivered.
- Interface. This field shows the name of the interface.
- Flags. This field defines up to five flags. Flags are on/off switches that signify either presence or absence. The five flags are U (up), G (gateway), H (host-specific), D (added by redirection), and M (modified by redirection).
  a. U (up). The U flag indicates the router is up and running. If this flag is not present, it means that the router is down. The packet cannot be forwarded and is discarded.
  b. G (gateway). The G flag means that the destination is in another network. The packet is delivered to the next-hop router for delivery (indirect delivery). When this flag is missing, it means the destination is in this network (direct delivery).

c. H (host-specific). The H flag indicates that the entry in the network address field is a host-specific address. When it is missing, it means that the address is only the network address of the destination.

d. D (added by redirection). The D flag indicates that routing information for this destination has been added to the host routing table by a redirection message from ICMP. We discussed redirection and the ICMP protocol in Chapter 21.

e. M (modified by redirection). The M flag indicates that the routing information for this destination has been modified by a redirection message from ICMP. We discussed redirection and the ICMP protocol in Chapter 21.

- Reference count. This field gives the number of users of this route at the moment. For example, if five people at the same time are connecting to the same host from this router, the value of this column is 5.
- Use. This field shows the number of packets transmitted through this router for the corresponding destination.

Utilities

There are several utilities that can be used to find the routing information and the contents of a routing table. We discuss *netstat* and *ifconfig.*

Example 22.6

One utility that can be used to find the contents of a routing table for a host or router is *netstat* in UNIX or LINUX. The following shows the list of the contents of a default server. We have used two options, r and n. The option r indicates that we are interested in the routing table, and the option n indicates that we are looking for numeric addresses. Note that this is a routing table for a host, not a router. Although we discussed the routing table for a router throughout the chapter, a host also needs a routing table.

```
$ netstat-rn
Kernel IP routing table
```

Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	ethO
127.0.0.0	0.0.0.0	255.0.0.0	U	10
0.0.0.0	153.18.31.254	0.0.0.0	UO	ethO

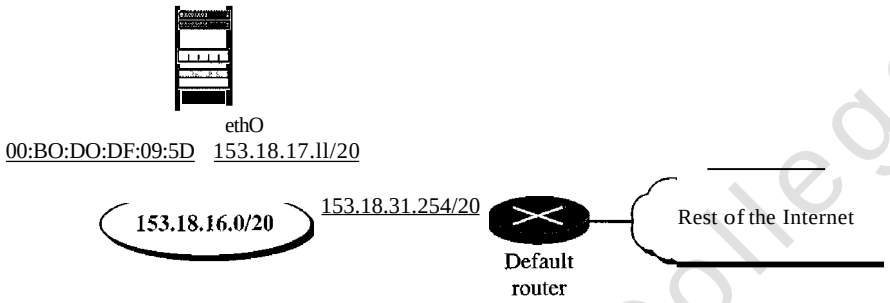
Note also that the order of columns is different from what we showed. The destination column here defines the network address. The term *gateway* used by UNIX is synonymous with *router.* This column acmally defines the address of the next hop. The value 0.0.0.0 shows that the delivery is direct. The last entry has a flag of G, which means that the destination can be reached through a router (default router). The *Iface* defines the interface. The host has only one real interface, ethO, which means interface 0 connected to an Ethernet network. The second interface, la, is actually a virmalloopback interface indicating that the host accepts packets with loopback address 127.0.0.0.

More information about the IP address and physical address of the server can be found by using the *ifconfig* command on the given interface (ethO).

```
$ ifconfig ethO
ethO   Link encap:Ethemet  HWaddr 00:BO:DO:DF:09:5D
inet addr:153.18.17.11  Bcast: 153.18.31.255  Mask:255.255.240.0
```

From the above information, we can deduce the configuration of the server, as shown in Figure 22.11.

Figure 22.11 *Configuration of the server for Example 22.6*



Note that the *ifconfig* command gives us the IP address and the physical (hardware) address of the interface.

22.3 UNICAST ROUTING PROTOCOLS

A routing table can be either static or dynamic. A *static table* is one with manual entries. A *dynamic table,* on the other hand, is one that is updated automatically when there is a change somewhere in the internet. Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a router is down, and they need to be updated whenever a better route has been found.

Routing protocols have been created in response to the demand for dynamic routing tables. A routing protocol is a combination of rules and procedures that lets routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhood. The sharing of information allows a router in San Francisco to know about the failure of a network in Texas. The routing protocols also include procedures for combining information received from other routers.

Optimization

A router receives a packet from a network and passes it to another network. A router is usually attached to several networks. When it receives a packet, to which network should it pass the packet? The decision is based on optimization: Which of the available pathways is the optimum pathway? What is the definition of the term *optimum?*

One approach is to assign a cost for passing through a network. We call this cost a metric. However, the metric assigned to each network depends on the type of protocol. Some simple protocols, such as the Routing Information Protocol (RIP), treat all networks as equals. The cost of passing through a network is the same; it is one hop count. So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts.

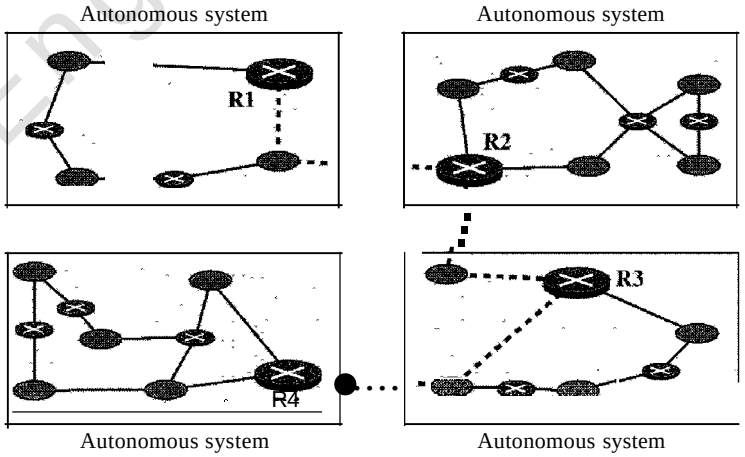
Other protocols, such as Open Shortest Path First (OSPF), allow the administrator to assign a cost for passing through a network based on the type of service required. A route through a network can have different costs (metrics). For example, if maximum throughput is the desired type of service, a satellite link has a lower metric than a fiber-optic line. On the other hand, if minimum delay is the desired type of service, a fiber-optic line has a lower metric than a satellite link. Routers use routing tables to help decide the best route. OSPF protocol allows each router to have several routing tables based on the required type of service.

Other protocols define the metric in a totally different way. In the Border Gateway Protocol (BGP), the criterion is the policy, which can be set by the administrator. The policy defines what paths should be chosen.

Intra- and Interdomain Routing

Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems. An autonomous system (AS) is a group of networks and routers under the authority of a single administration. Routing inside an autonomous system is referred to as intradomain routing. Routing between autonomous systems is referred to as interdomain routing. Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system. However, only one interdomain routing protocol handles routing between autonomous systems (see Figure 22.12).

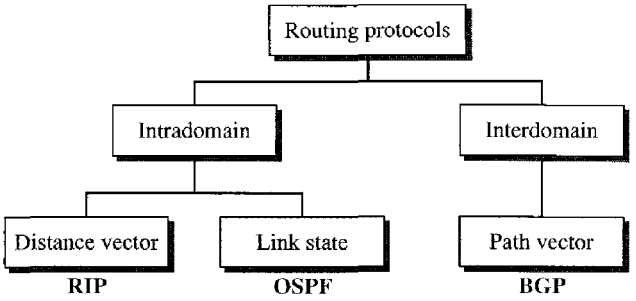
Figure 22.12 *Autonomous systems*



Several intradomain and interdomain routing protocols are in use. In this section, we cover only the most popular ones. We discuss two intradomain routing protocols: distance vector and link state. We also introduce one interdomain routing protocol: path vector (see Figure 22.13).

Routing Information Protocol (RIP) is an implementation of the distance vector protocol. Open Shortest Path First (OSPF) is an implementation of the link state protocol. Border Gateway Protocol (BGP) is an implementation of the path vector protocol.

Figure 22.13 *Popular routing protocols*



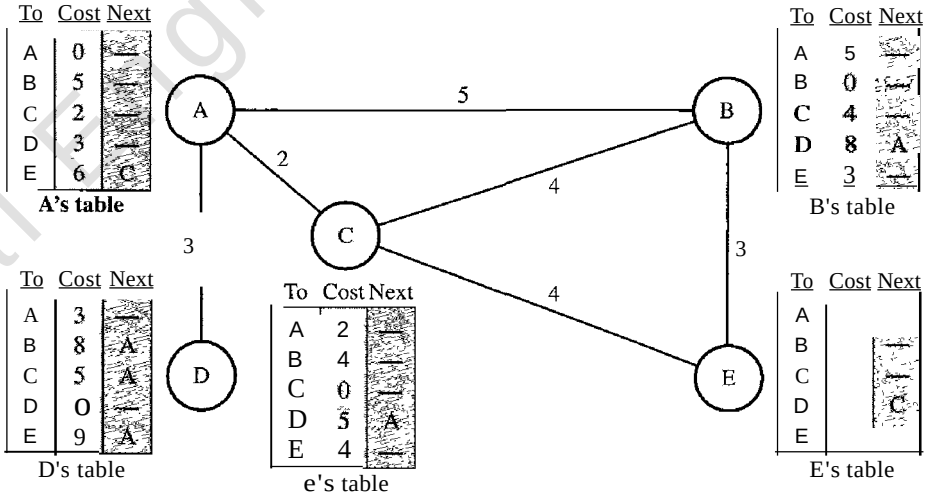
Distance Vector Routing

In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing).

We can think of nodes as the cities in an area and the lines as the roads connecting them. A table can show a tourist the minimum distance between cities.

In Figure 22.14, we show a system of five nodes with their corresponding tables.

Figure 22.14 *Distance vector routing tables*



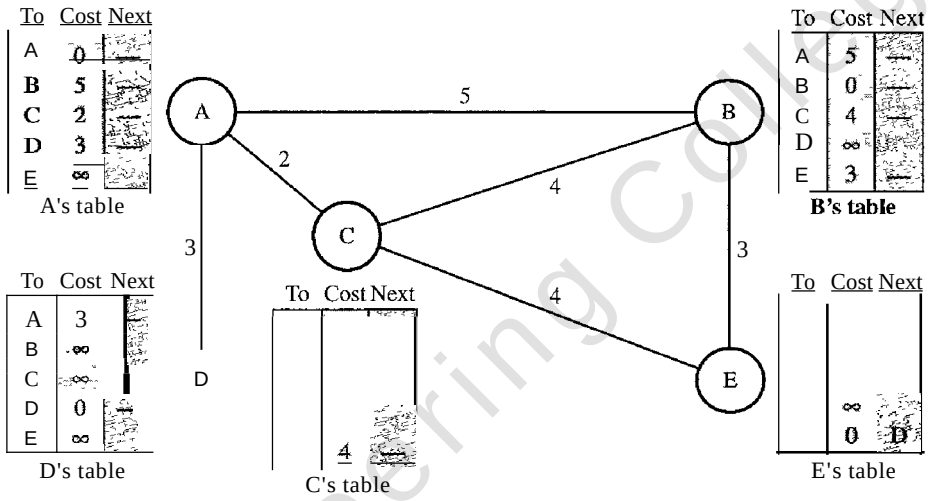
The table for node A shows how we can reach any node from this node. For example, our least cost to reach node E is 6. The route passes through C.

Initialization

The tables in Figure 22.14 are stable; each node knows how to reach any other node and the cost. At the beginning, however, this is not the case. Each node can know only

the distance between itself and its immediate neighbors, those directly connected to it. So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors. Figure 22.15 shows the initial tables for each node. The distance for any entry that is not a neighbor is marked as infinite (unreachable).

Figure 22.15 *Initialization of tables in distance vector routing*



Sharing

The whole idea of distance vector routing is the sharing of information between neighbors. Although node A does not know about node E, node C does. So if node C shares its routing table with A, node A can also know how to reach node E. On the other hand, node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.

There is only one problem. How much of the table must be shared with each neighbor? A node is not aware of a neighbor's table. The best solution for each node is to send its entire table to the neighbor and let the neighbor decide what part to use and what part to discard. However, the third column of a table (next stop) is not useful for the neighbor. When the neighbor receives a table, this column needs to be replaced with the sender's name. If any of the rows can be used, the next node is the sender of the table. A node therefore can send only the first two columns of its table to any neighbor. In other words, sharing here means sharing only the first two columns.

In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

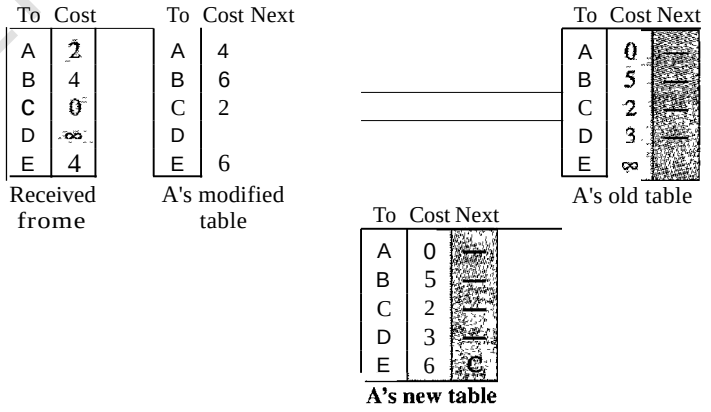
Updating

When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is x mi, and the distance between A and C is y mi, then the distance between A and that destination, via C, is $x + y$ mi.
2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
   a. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is a tie, the old one is kept.
   b. If the next-node entry is the same, the receiving node chooses the new row. For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

Figure 22.16 shows how node A updates its routing table after receiving the partial table from node C.

Figure 22.16 *Updating in distance vector routing*



There are several points we need to emphasize here. First, as we know from mathematics, when we add any number to infinity, the result is still infinity. Second, the modified table shows how to reach A from A via C. If A needs to reach itself via C, it needs to go to C and come back, a distance of 4. Third, the only benefit from this updating of node A is the last entry, how to reach E. Previously, node A did not know how to reach E (distance of infinity); now it knows that the cost is 6 via C.

Each node can update its table by using the tables received from other nodes. In a short time, if there is no change in the network itself, such as a failure in a link, each node reaches a stable condition in which the contents of its table remains the same.

When to Share

The question now is, When does a node send its partial routing table (only two columns) to all its immediate neighbors? The table is sent both periodically and when there is a change in the table.

Periodic Update A node sends its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

Triggered Update A node sends its two-column routing table to its neighbors anytime there is a change in its routing table. This is called a triggered update. The change can result from the following.

1. A node receives a table from a neighbor, resulting in changes in its own table after updating.
2. A node detects some failure in the neighboring links which results in a distance change to infinity.

Two-Node Loop Instability

A problem with distance vector routing is instability, which means that a network using this protocol can become unstable. To understand the problem, let us look at the scenario depicted in Figure 22.17.

Figure 22.17 *Two-node instability*

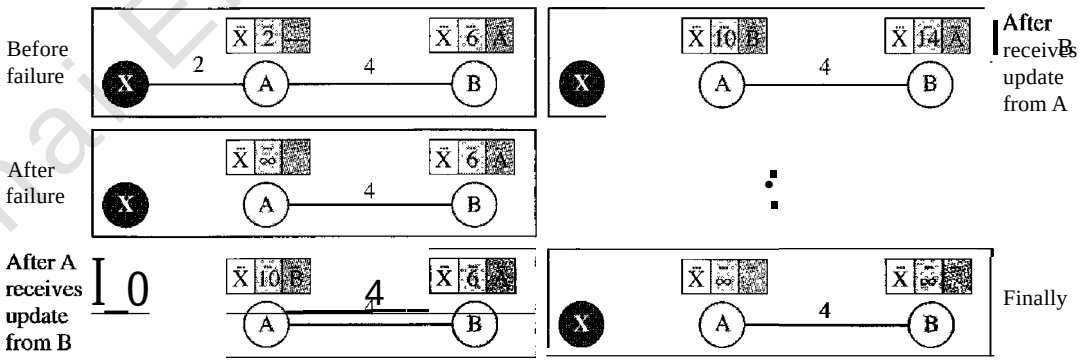


Figure 22.17 shows a system with three nodes. We have shown only the portions of the routing table needed for our discussion. At the beginning, both nodes A and B know how to reach node X. But suddenly, the link between A and X fails. Node A changes its table. If A can send its table to B immediately, everything is fine. However, the system becomes unstable if B sends its routing table to A before receiving A's routing table. Node A receives the update and, assuming that B has found a way to reach X, immediately updates its routing table. Based on the triggered update strategy, A sends its new

update to B. Now B thinks that something has been changed around A and updates its routing table. The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached. However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A receives a packet destined for X, it goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem. A few solutions have been proposed for instability of this kind.

Defining Infinity The first obvious solution is to redefine infinity to a smaller number, such as 100. For our previous scenario, the system will be stable in less than 20 updates. As a matter of fact, most implementations of the distance vector protocol define the distance between each node to be I and define 16 as infinity. However, this means that the distance vector routing cannot be used in large systems. The size of the network, in each direction, can not exceed 15 hops.

Split Horizon Another solution is called split horizon. In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has corne from A (A already knows). Taking information from node A, modifying it, and sending it back to node A creates the confusion. In our scenario, node B eliminates the last line of its routing table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later when node A sends its routing table to B, node B also corrects its routing table. The system becomes stable after the first update: both node A and B know that X is not reachable.

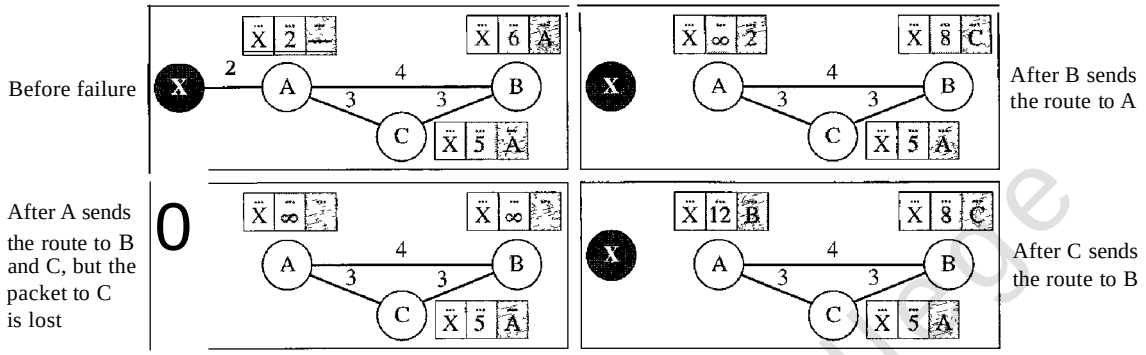
Split Horizon and Poison Reverse Using the split horizon strategy has one drawback. Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently. The split horizon strategy can be combined with the poison reverse strategy. Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

Three-Node Instability

The two-node instability can be avoided by using the split horizon strategy combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed. Figure 22.18 shows the scenario.

Suppose, after finding that X is not reachable, node A sends a packet to Band C to inform them of the situation. Node B immediately updates its table, but the packet to C is lost in the network and never reaches C. Node C remains in the dark and still thinks that there is a route to X via A with a distance of 5. After a while, node C sends to Bits routing table, which includes the route to X. Node B is totally fooled here. It receives information on the route to X from C, and according to the algorithm, it updates its

Figure 22.18 *Three-node instability*



table, showing the route to X via C with a cost of 8. This information has come from C, not from A, so after awhile node B may advertise this route to A. Now A is fooled and updates its table to show that A can reach X via B with a cost of 12. Of course, the loop continues; now A advertises the route to X to C, with increased cost, but not to B. Node C then advertises the route to B with an increased cost. Node B does the same to A. And so on. The loop stops when the cost in each node reaches infinity.

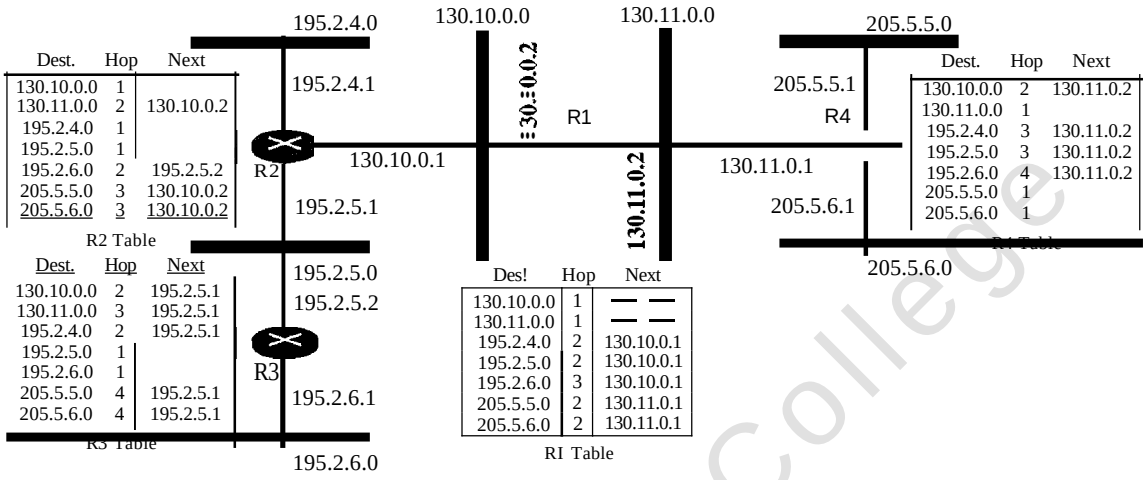
RIP

The Routing Information Protocol (RIP) is an intradomain routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with routers and networks (links). The routers have routing tables; networks do not.
2. The destination in a routing table is a network, which means the first column defines a network address.
3. The metric used by RIP is very simple; the distance is defined as the number of links (networks) to reach the destination. For this reason, the metric in RIP is called a hop count.
4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
5. The next-node column defines the address of the router to which the packet is to be sent to reach its destination.

Figure 22.19 shows an autonomous system with seven networks and four routers. The table of each router is also shown. Let us look at the routing table for R1. The table has seven entries to show how to reach each network in the autonomous system. Router R1 is directly connected to networks 130.10.0.0 and 130.11.0.0, which means that there are no next-hop entries for these two networks. To send a packet to one of the three networks at the far left, router R1 needs to deliver the packet to R2. The next-node entry for these three networks is the interface of router R2 with IP address 130.10.0.1. To send a packet to the two networks at the far right, router R1 needs to send the packet to the interface of router R4 with IP address 130.11.0.1. The other tables can be explained similarly.

Figure 22.19 *Example of a domain using RIP*



Dest.	Hop	Next
130.10.0.0	1	
130.11.0.0	2	130.10.0.2
195.2.4.0	1	
195.2.5.0	1	
195.2.6.0	2	195.2.5.2
205.5.5.0	3	130.10.0.2
205.5.6.0	3	130.10.0.2

Dest.	Hop	Next
130.10.0.0	2	195.2.5.1
130.11.0.0	3	195.2.5.1
195.2.4.0	2	195.2.5.1
195.2.5.0	1	
195.2.6.0	1	
205.5.5.0	4	195.2.5.1
205.5.6.0	4	195.2.5.1

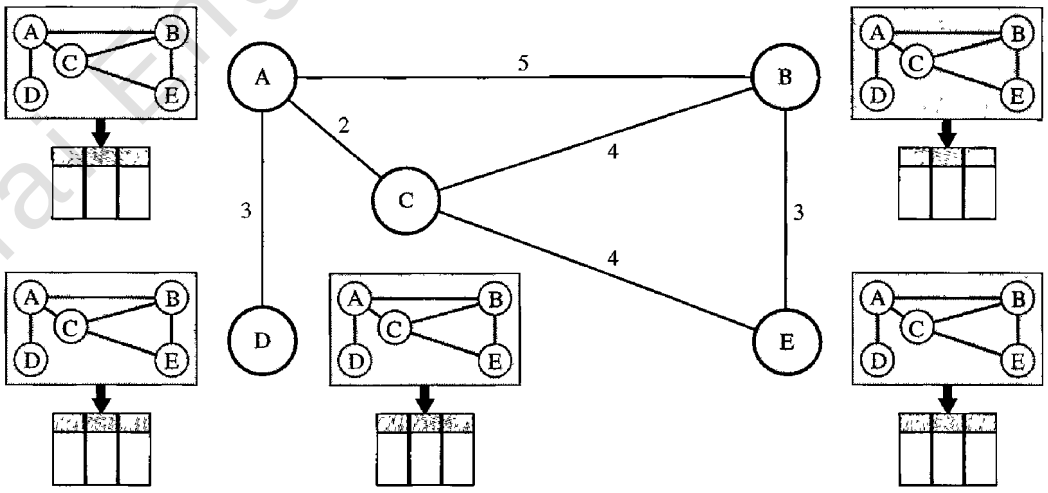
Des!	Hop	Next
130.10.0.0	1	— —
130.11.0.0	1	— —
195.2.4.0	2	130.10.0.1
195.2.5.0	2	130.10.0.1
195.2.6.0	3	130.10.0.1
205.5.5.0	2	130.11.0.1
205.5.6.0	2	130.11.0.1

Dest.	Hop	Next
130.10.0.0	2	130.11.0.2
130.11.0.0	1	
195.2.4.0	3	130.11.0.2
195.2.5.0	3	130.11.0.2
195.2.6.0	4	130.11.0.2
205.5.5.0	1	
205.5.6.0	1	

Link State Routing

Link state routing has a different philosophy from that of distance vector routing. In link state routing, if each node in the domain has the entire topology of the domain-the list of nodes and links, how they are connected including the type, cost (metric), and condition of the links (up or down)-the node can use Dijkstra's algorithm to build a routing table. Figure 22.20 shows the concept.

Figure 22.20 *Concept of link state routing*

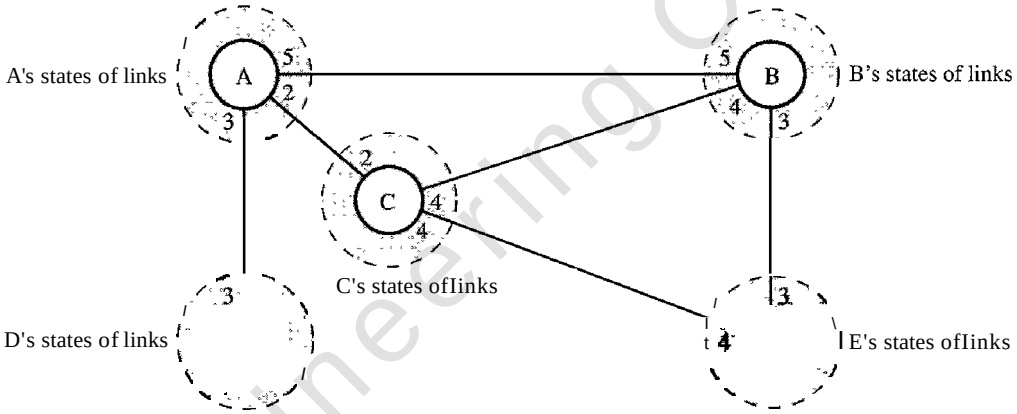


The figure shows a simple domain with five nodes. Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology. This is analogous to a city map. While each person may have the same map, each needs to take a different route to reach her specific destination.

The topology must be dynamic, representing the latest state of each node and each link. If there are changes in any point in the network (a link is down, for example), the topology must be updated for each node.

How can a common topology be dynamic and stored in each node? No node can know the topology at the beginning or after a change somewhere in the network. Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links. In other words, the whole topology can be compiled from the partial knowledge of each node. Figure 22.21 shows the same domain as in Figure 22.20, indicating the part of the knowledge belonging to each node.

Figure 22.21 *Link state knowledge*



Node A knows that it is connected to node B with metric 5, to node C with metric 2, and to node D with metric 3. Node C knows that it is connected to node A with metric 2, to node B with metric 4, and to node E with metric 4. Node D knows that it is connected only to node A with metric 3. And so on. Although there is an overlap in the knowledge, the overlap guarantees the creation of a common topology-a picture of the whole domain for each node.

Building Routing Tables

In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

]. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding,** in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP) A link state packet can carry a large amount of information. For the moment, however, we assume that it carries a minimum amount

of data: the node identity, the list of links, a sequence number, and age. The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from remaining in the domain for a long time. LSPs are generated on two occasions:

1. *When there is a change in the topology of the domain.* Triggering of LSP dissemination is the main way of quickly informing any node in the domain to update its topology.
2. *On a periodic basis.* The period in this case is much longer compared to distance vector routing. As a matter of fact, there is no actual need for this type of LSP dissemination. It is done to ensure that old information is removed from the domain. The timer set for periodic dissemination is normally in the range of 60 min or 2 h based on the implementation. A longer period ensures that flooding does not create too much traffic on the network.

Flooding of LSPs After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding and based on the following:

1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:
   a. It discards the old LSP and keeps the new one.
   b. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

Formation of Shortest Path Tree: Dijkstra Algorithm After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.

A tree is a graph of nodes and links; one node is called the root. All other nodes can be reached from the root through only one single route. A shortest path tree is a tree in which the path between the root and every other node is the shortest. What we need for each node is a shortest path tree with that node as the root.

The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: tentative and permanent. It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent. We can informally define the algorithm by using the flowchart in Figure 22.22.

Let us apply the algorithm to node A of our sample graph in Figure 22.23. To find the shortest path in each step, we need the cumulative cost from the root to each node, which is shown next to the node.

The following shows the steps. At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.

Figure 22.22 *Dijkstra algorithm*

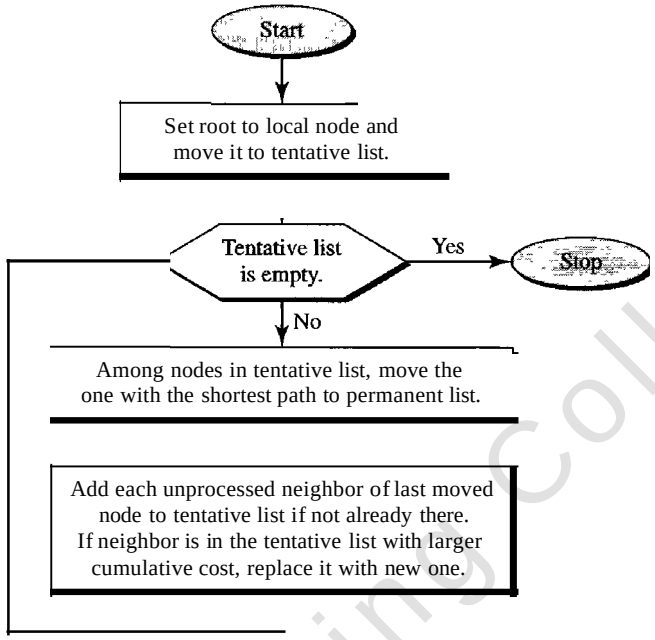
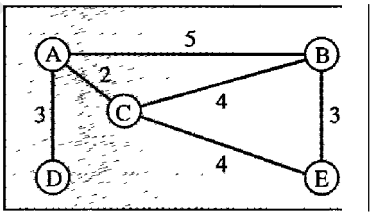
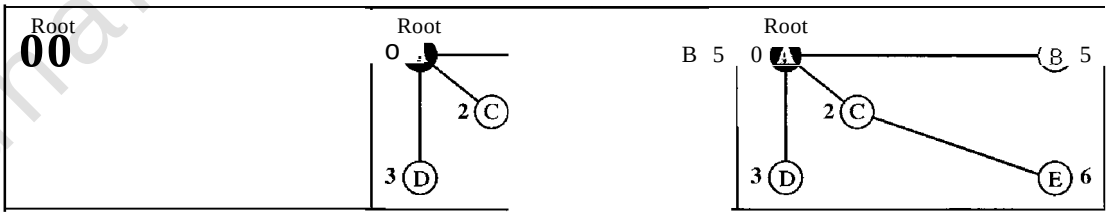


Figure 22.23 *Example offormation ofshortest path tree*



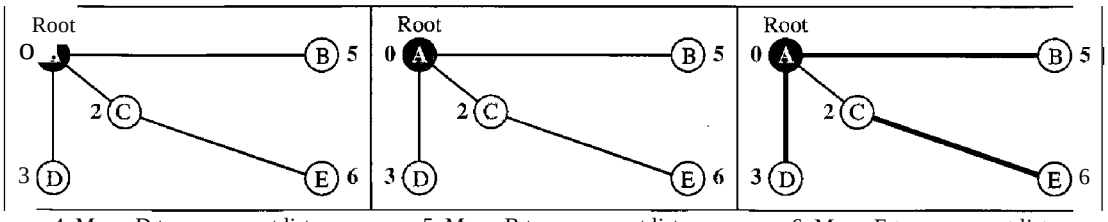
Topology



1. Set root to A and move A to tentative list.

2. Move A to permanent list and add B, C, and D to tentative list.

3. Move C to permanent and add E to tentative list.



4. Move D to permanent list.

5. Move B to permanent list.

6. Move E to permanent list (tentative list is empty).

1. We make node A the root of the tree and move it to the tentative list. Our two lists are

   Permanent list: empty Tentative list: A(O)

2. Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to the pennanent list and add all neighbors of A to the tentative list. Our new lists are

   Permanent list: A(O) Tentative list: B(5), C(2), D(3)

3. Node C has the shortest cumulative cost from all nodes in the tentative list. We move C to the permanent list. Node C has three neighbors, but node A is already processed, which makes the unprocessed neighbors just B and E. However, B is already in the tentative list with a cumulative cost of 5. Node A could also reach node B through C with a cumulative cost of 6. Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it. Our new lists are

   Permanent list: A(O), e(2) Tentative list: B(5), 0(3), E(6)

4. Node D has the shortest cumulative cost of all the nodes in the tentative list. We move D to the permanent list. Node D has no unprocessed neighbor to be added to the tentative list. Our new lists are

   Permanent list: A(O), C(2), 0(3) Tentative list: B(5), E(6)

5. Node B has the shortest cumulative cost of all the nodes in the tentative list. We move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (this is just node E). However, E(6) is already in the list with a smaller cumulative cost. The cumulative cost to node E, as the neighbor of B, is 8. We keep node E(6) in the tentative list. Our new lists are

   Permanent list: A(O), B(5), C(2), 0(3) Tentative list: E(6)

6. Node E has the shortest cumulative cost from all nodes in the tentative list. We move E to the permanent list. Node E has no neighbor. Now the tentative list is empty. We stop; our shortest path tree is ready. The final lists are

   Permanent list: A(0), B(5), C(2), D(3), E(6) Tentative list: empty

Calculation of Routing Table from Shortest Path Tree Each node uses the shortest path tree protocol to construct its routing table. The routing table shows the cost of reaching each node from the root. Table 22.2 shows the routing table for node A.

Table 22.2 *Routing table for node A*

Node	*Cost*	*Next Router*
A	0	-
B	5	-
C	2	-
D	3	-
E	6	C

Compare Table 22.2 with the one in Figure 22.14. Both distance vector routing and link state routing end up with the same routing table for node A.

OSPF

The Open Shortest Path First or OSPF protocol is an intradomain routing protocol based on link state routing. Its domain is also an autonomous system.

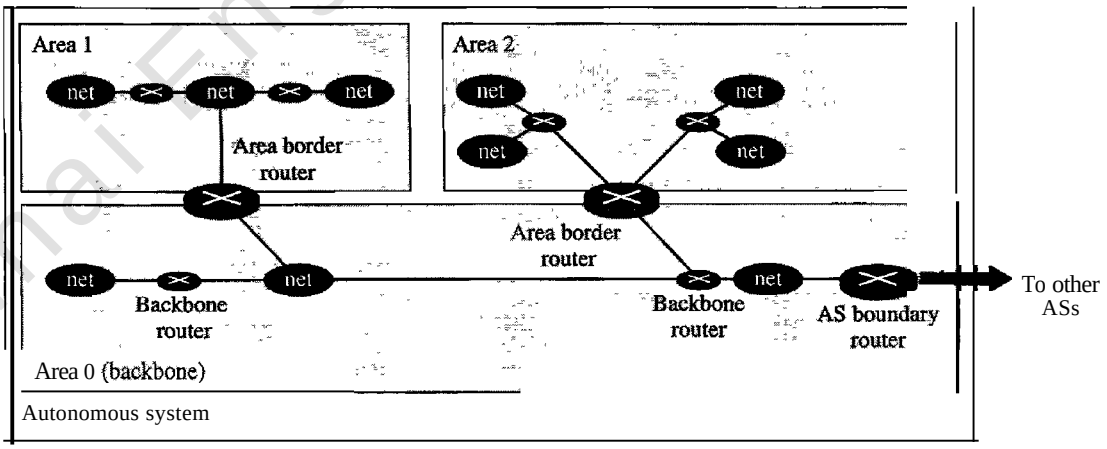
Areas To handle routing efficiently and in a timely manner, OSPF divides an autonomous system into areas. An area is a collection of networks, hosts, and routers all contained within an autonomous system. An autonomous system can be divided into many different areas. All networks inside an area must be connected.

Routers inside an area flood the area with routing information. At the border of an area, special routers called area border routers summarize the information about the area and send it to other areas. Among the areas inside an autonomous system is a special area called the *backbone;* all the areas inside an autonomous system must be connected to the backbone. In other words, the backbone serves as a primary area and the other areas as secondary areas. This does not mean that the routers within areas cannot be connected to each other, however. The routers inside the backbone are called the backbone routers. Note that a backbone router can also be an area border router.

If, because of some problem, the connectivity between a backbone and an area is broken, a virtual link between routers must be created by an administrator to allow continuity of the functions of the backbone as the primary area.

Each area has an area identification. The area identification of the backbone is zero. Figure 22.24 shows an autonomous system and its areas.

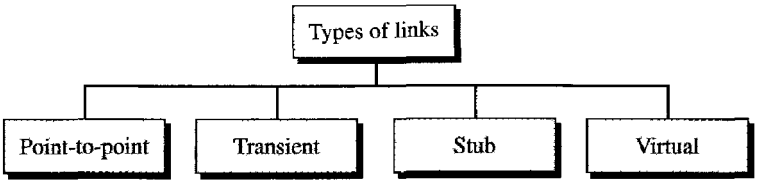
Figure 22.24 *Areas in an autonomous system*



Metric The OSPF protocol allows the administrator to assign a cost, called the metric, to each route. The metric can be based on a type of service (minimum delay, maximum throughput, and so on). As a matter of fact, a router can have multiple routing tables, each based on a different type of service.

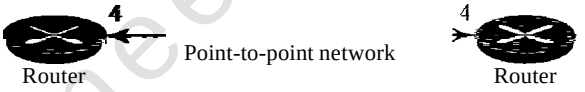
Types of Links In OSPF terminology, a connection is called a *link.* Four types of links have been defined: point-to-point, transient, stub, and virtual (see Figure 22.25).

Figure 22.25 *Types of links*



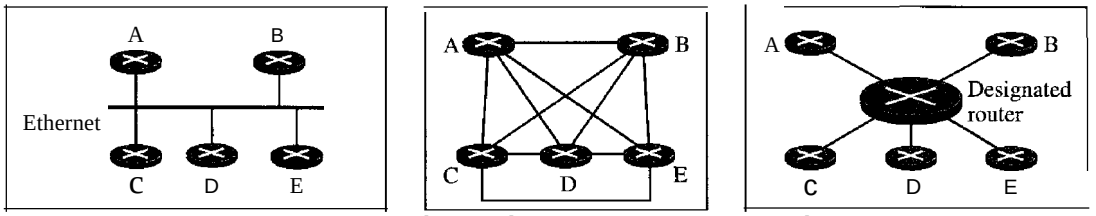
A point-to-point link connects two routers without any other host or router in between. In other words, the purpose of the link (network) is just to connect the two routers. An example of this type of link is two routers connected by a telephone line or a T line. There is no need to assign a network address to this type of link. Graphically, the routers are represented by nodes, and the link is represented by a bidirectional edge connecting the nodes. The metrics, which are usually the same, are shown at the two ends, one for each direction. In other words, each router has only one neighbor at the other side of the link (see Figure 22.26).

Figure 22.26 *Point-to-point link*



A transient link is a network with several routers attached to it. The data can enter through any of the routers and leave through any router. All LANs and some WANs with two or more routers are of this type. In this case, each router has many neighbors. For example, consider the Ethernet in Figure 22.27a. Router A has routers B, C, D, and E as neighbors. Router B has routers A, C, D, and E as neighbors. If we want to show the neighborhood relationship in this situation, we have the graph shown in Figure 22.27b.

Figure 22.27 *Transient link*



a. Transient network b. Unrealistic representation c. Realistic representation

This is neither efficient nor realistic. It is not efficient because each router needs to advertise the neighborhood to four other routers, for a total of 20 advertisements. It is

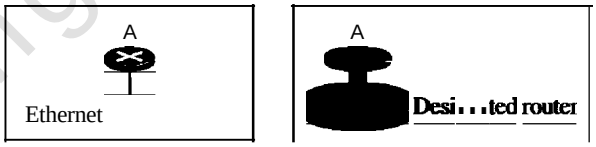
not realistic because there is no single network (link) between each pair of routers; there is only one network that serves as a crossroad between all five routers.

To show that each router is connected to every other router through one single network, the network itself is represented by a node. However, because a network is not a machine, it cannot function as a router. One of the routers in the network takes this responsibility. It is assigned a dual purpose; it is a true router and a designated router. We can use the topology shown in Figure 22.27c to show the connections of a transient network.

Now each router has only one neighbor, the designated router (network). On the other hand, the designated router (the network) has five neighbors. We see that thc number of neighbor announcements is reduced from 20 to 10. Still, the link is represented as a bidirectional edge between the nodes. However, while there is a metric from each node to the designated router, there is no metric from the designated router to any other node. The reason is that the designated router represents the network. We can only assign a cost to a packet that is passing through the network. We cannot charge for this twice. When a packet enters a network, we assign a cost; when a packet leaves the network to go to the router, there is no charge.

A **stub link** is a network that is connected to only one router. The data packets enter the network through this single router and leave the network through this same router. This is a special case of the transient network. We can show this situation using the router as a node and using the designated router for the network. However, the link is only one-directional, from the router to the network (see Figure 22.28).

Figure 22.28 *Stub link*



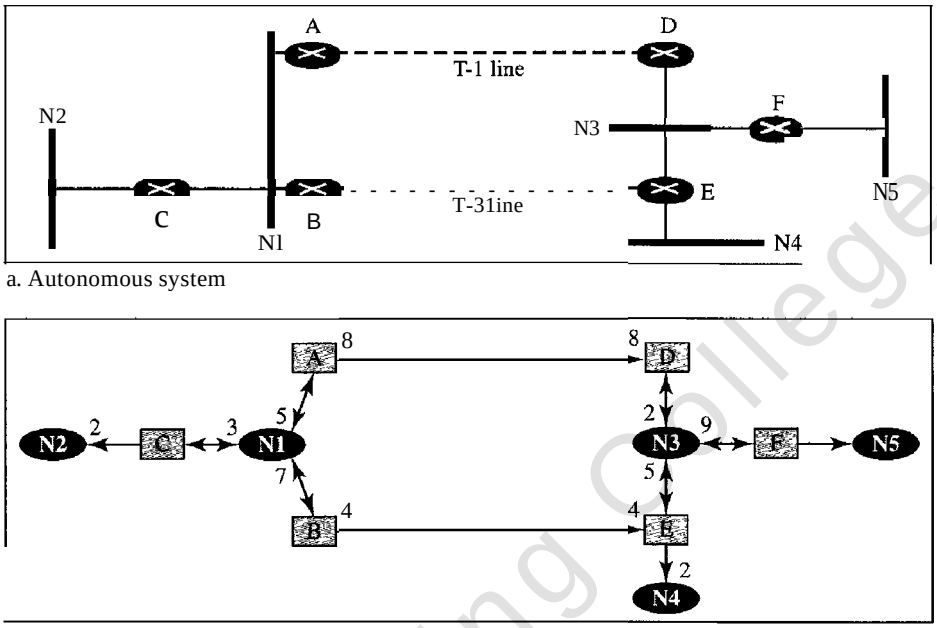
a. Stub network b. Representation

When the link between two routers is broken, the administration may create a **virtual link** between them, using a longer path that probably goes through several routers.

Graphical Representation Let us now examine how an AS can be represented graphically. Figure 22.29 shows a small AS with seven networks and six routers. Two of the networks are point-to-point networks. We use symbols such as Nl and N2 for transient and stub networks. There is no need to assign an identity to a point-to-point network. The figure also shows the graphical representation of the AS as seen by OSPF.

We have used square nodes for the routers and ovals for the networks (represented by designated routers). However, OSPF sees both as nodes. Note that we have three stub networks.

Figure 22.29 *Example of an AS and its graphical representation in OSPF*



Path Vector Routing

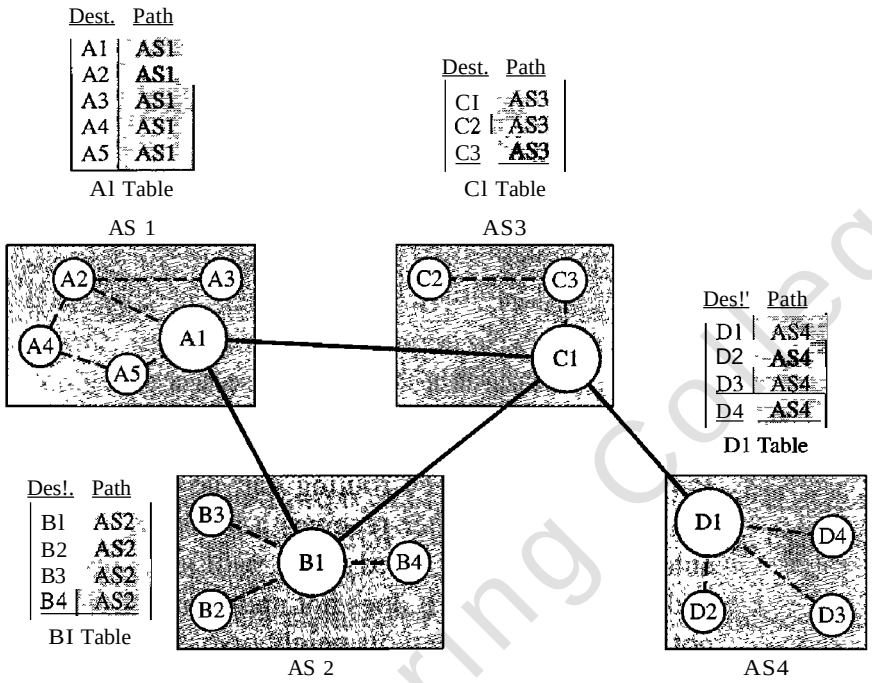
Distance vector and link state routing are both intradomain routing protocols. They can be used inside an autonomous system, but not between autonomous systems. These two protocols are not suitable for interdomain routing mostly because of scalability. Both of these routing protocols become intractable when the domain of operation becomes large. Distance vector routing is subject to instability if there are more than a few hops in the domain of operation. Link state routing needs a huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding. There is a need for a third routing protocol which we call path vector routing.

Path vector routing proved to be useful for interdomain routing. The principle of path vector routing is similar to that of distance vector routing. In path vector routing, we assume that there is one node (there can be more, but one is enough for our conceptual discussion) in each autonomous system that acts on behalf of the entire autonomous system. Let us call it the speaker node. The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs. The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other. However, what is advertised is different. A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems.

Initialization

At the beginning, each speaker node can know only the reachability of nodes inside its autonomous system. Figure 22.30 shows the initial tables for each speaker node in a system made of four ASs.

Figure 22.30 *Initial routing tables in path vector routing*



Node Al is the speaker node for ASl, Bl for AS2, Cl for AS3, and Dl for AS4. Node Al creates an initial table that shows Al to A5 are located in ASI and can be reached through it. Node Bl advertises that Bl to B4 are located in AS2 and can be reached through Bl. And so on.

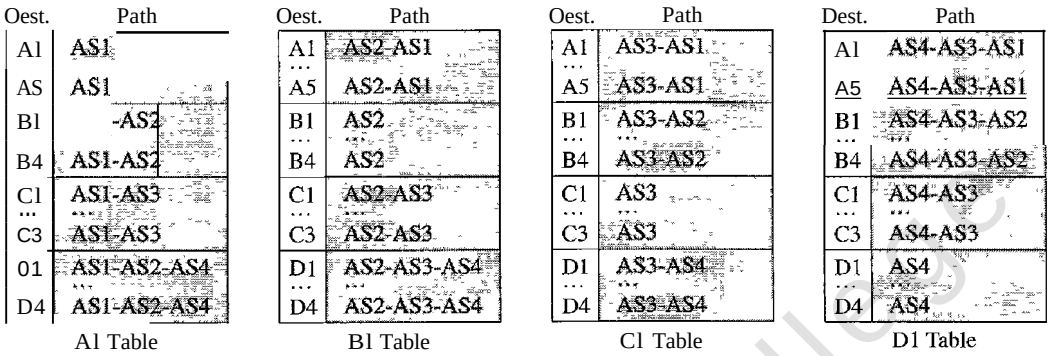
Sharing Just as in distance vector routing, in path vector routing, a speaker in an autonomous system shares its table with immediate neighbors. In Figure 22.30, node Al shares its table with nodes Bl and Cl. Node Cl shares its table with nodes Dl, Bl, andAl. Node Bl shares its table with Cl andAl. Node Dl shares its table with Cl.

Updating When a speaker node receives a two-column table from a neighbor, it updates its own table by adding the nodes that are not in its routing table and adding its own autonomous system and the autonomous system that sent the table. After a while each speaker has a table and knows how to reach each node in other ASs. Figure 22.31 shows the tables for each speaker node after the system is stabilized.

According to the figure, if router Al receives a packet for nodes A3, it knows that the path is in ASI (the packet is at home); but if it receives a packet for Dl, it knows that the packet should go from ASl, to AS2, and then to AS3. The routing table shows the path completely. On the other hand, if node Dl in AS4 receives a packet for node A2, it knows it should go through AS4, AS3, and AS 1.

- Loop prevention. The instability of distance vector routing and the creation of loops can be avoided in path vector routing. When a router receives a message, it checks to see if its autonomous system is in the path list to the destination. If it is, looping is involved and the message is ignored.

Figure 22.31 *Stabilized tables for three autonomous systems*



- **Policy routing.** Policy routing can be easily implemented through path vector routing. When a router receives a message, it can check the path. If one of the autonomous systems listed in the path is against its policy, it can ignore that path and that destination. It does not update its routing table with this path, and it does not send this message to its neighbors.
- **Optimum path.** What is the optimum path in path vector routing? We are looking for a path to a destination that is the best for the organization that runs the autonomous system. We definitely cannot include metrics in this route because each autonomous system that is included in the path may use a different criterion for the metric. One system may use, internally, RIP, which defines hop count as the metric; another may use OSPF with minimum delay defined as the metric. The optimum path is the path that fits the organization. In our previous figure, each autonomous system may have more than one path to a destination. For example, a path from AS4 to ASI can be AS4-AS3-AS2-AS1, or it can be AS4-AS3-ASI. For the tables, we chose the one that had the smaller number of autonomous systems, but this is not always the case. Other criteria, such as security, safety, and reliability, can also be applied.

BGP

Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. It first appeared in 1989 and has gone through four versions.

Types **of Autonomous** Systems As we said before, the Internet is divided into hierarchical domains called autonomous systems. For example, a large corporation that manages its own network and has full control over it is an autonomous system. A local ISP that provides services to local customers is an autonomous system. We can divide autonomous systems into three categories: stub, multihomed, and transit.

- **Stub** AS. A stub AS has only one connection to another AS. The interdomain data traffic in a stub AS can be either created or terminated in the AS. The hosts in the AS can send data traffic to other ASs. The hosts in the AS can receive data coming from hosts in other ASs. Data traffic, however, cannot pass through a stub AS. A stub AS

is either a source or a sink. A good example of a stub AS is a small corporation or a small local ISP.

- Multihomed AS. A multihomed AS has more than one connection to other ASs, but it is still only a source or sink for data traffic. It can receive data traffic from more than one AS. It can send data traffic to more than one AS, but there is no transient traffic. It does not allow data coming from one AS and going to another AS to pass through. A good example of a multihomed AS is a large corporation that is connected to more than one regional or national AS that does not allow transient traffic.
- Transit AS. A transit AS is a multihomed AS that also allows transient traffic. Good examples of transit ASs are national and international ISPs (Internet backbones).

Path Attributes In our previous example, we discussed a path for a destination network. The path was presented as a list of autonomous systems, but is, in fact, a list of attributes. Each attribute gives some information about the path. The list of attributes helps the receiving router make a more-informed decision when applying its policy.

Attributes are divided into two broad categories: well known and optional. A well-known attribute is one that every BGP router must recognize. An optional attribute is one that needs not be recognized by every router.

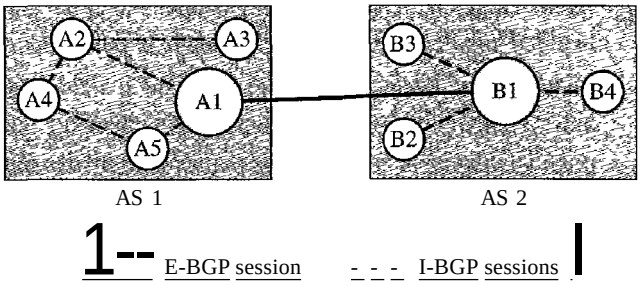
Well-known attributes are themselves divided into two categories: mandatory and discretionary. A *well-known mandatory attribute* is one that must appear in the description of a route. A *well-known discretionary attribute* is one that must be recognized by each router, but is not required to be included in every update message. One well-known mandatory attribute is ORIGIN. This defines the source of the routing information (RIP, OSPF, and so on). Another well-known mandatory attribute is AS_PATH. This defines the list of autonomous systems through which the destination can be reached. Still another well-known mandatory attribute is NEXT-HOP, which defines the next router to which the data packet should be sent.

The optional attributes can also be subdivided into two categories: transitive and nontransitive. An *optional transitive attribute* is one that must be passed to the next router by the router that has not implemented this attribute. An *optional nontransitive attribute* is one that must be discarded if the receiving router has not implemented it.

BGP Sessions The exchange of routing information between two routers using BGP takes place in a session. A session is a connection that is established between two BGP routers only for the sake of exchanging routing information. To create a reliable environment, BGP uses the services of TCP. In other words, a session at the BGP level, as an application program, is a connection at the TCP level. However, there is a subtle difference between a connection in TCP made for BGP and other application programs. When a TCP connection is created for BGP, it can last for a long time, until something unusual happens. For this reason, BGP sessions are sometimes referred to as *semi-pennanent connections.*

External and Internal BGP If we want to be precise, BGP can have two types of sessions: external BGP (E-BGP) and internal BGP (I-BGP) sessions. The E-BGP session is used to exchange information between two speaker nodes belonging to two different autonomous systems. The I-BGP session, on the other hand, is used to exchange routing information between two routers inside an autonomous system. Figure 22.32 shows the idea.

Figure 22.32 *Internal and external BGP sessions*



The session established between AS 1 and AS2 is an E-BOP session. The two speaker routers exchange information they know about networks in the Internet. However, these two routers need to collect information from other routers in the autonomous systems. This is done using I-BOP sessions.

22.4 MULTICAST ROUTING PROTOCOLS

In this section, we discuss multicasting and multicast routing protocols. We first define the term *multicasting* and compare it to unicasting and broadcasting. We also briefly discuss the applications of multicasting. Finally, we move on to multicast routing and the general ideas and goals related to it. We also discuss some common multicast routing protocols used in the Internet today.

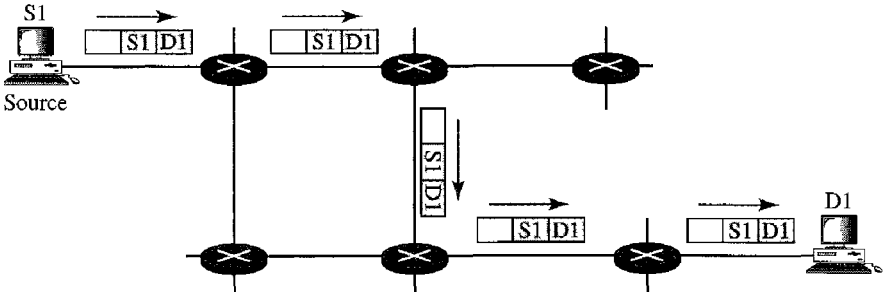
Unicast, Multicast, **and** Broadcast

A message can be unicast, multicast, or broadcast. Let us clarify these terms as they relate to the Internet.

Unicasting

In unicast communication, there is one source and one destination. The relationship between the source and the destination is one-to-one. In this type of communication, both the source and destination addresses, in the IP datagram, are the unicast addresses assigned to the hosts (or host interfaces, to be more exact). In Figure 22.33, a unicast

Figure 22.33 *Unicasting*



packet starts from the source S1 and passes through routers to reach the destination D1. We have shown the networks as a link between the routers to simplify the figure.

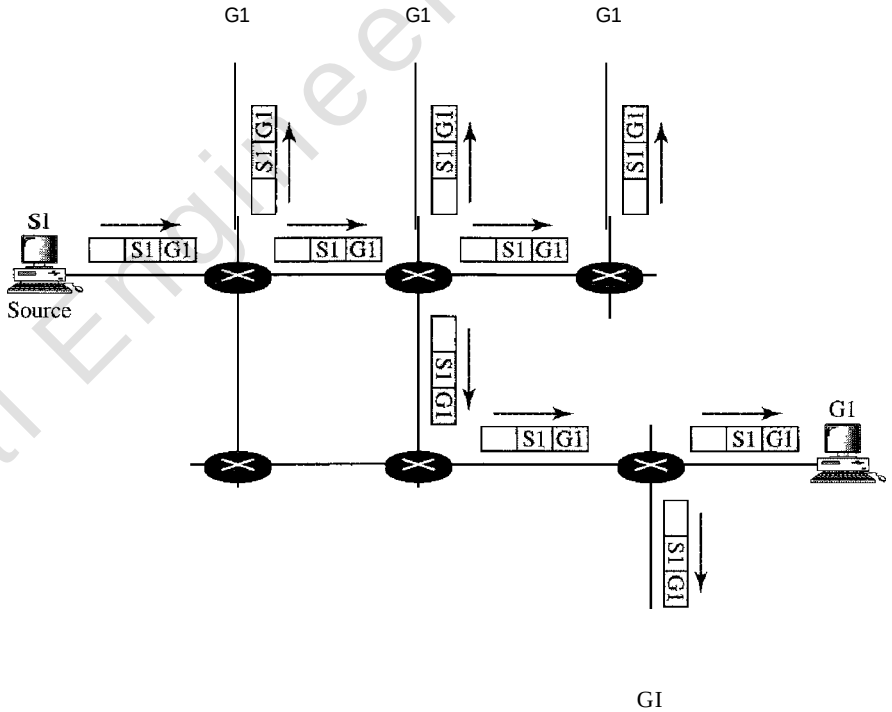
Note that in unicasting, when a router receives a packet, it forwards the packet through only one of its interfaces (the one belonging to the optimum path) as defined in the routing table. The router may discard the packet if it cannot find the destination address in its routing table.

In unicasting, the router forwards the received packet through only one of its interfaces.

Multicasting

In multicast communication, there is one source and a group of destinations. The relationship is one-to-many. In this type of communication, the source address is a unicast address, but the destination address is a group address, which defines one or more destinations. The group address identifies the members of the group. Figure 22.34 shows the idea behind multicasting.

Figure 22.34 *Multicasting*



A multicast packet starts from the source S1 and goes to all destinations that belong to group G1. In multicasting, when a router receives a packet, it may forward it through several of its interfaces.

In multicasting, the router may forward the received packet through several of its interfaces.

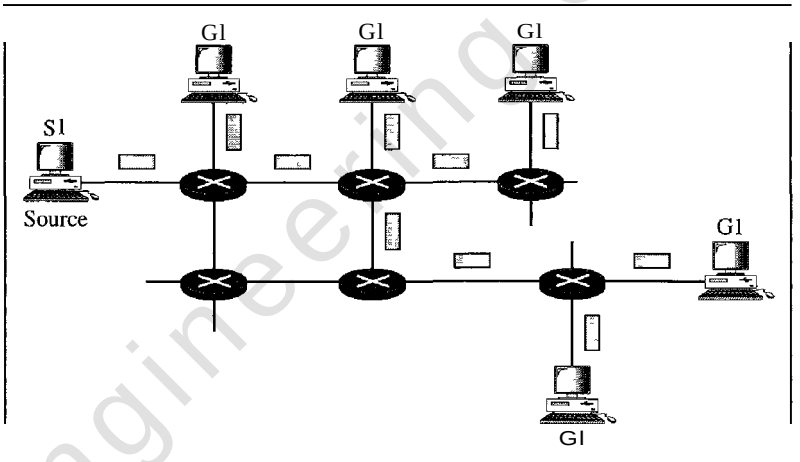
Broadcasting

In broadcast communication, the relationship between the source and the destination is one-to-all. There is only one source, but all the other hosts are the destinations. The Internet does not explicitly support broadcasting because of the huge amount of traffic it would create and because of the bandwidth it would need. Imagine the traffic generated in the Internet if one person wanted to send a message to everyone else connected to the Internet.

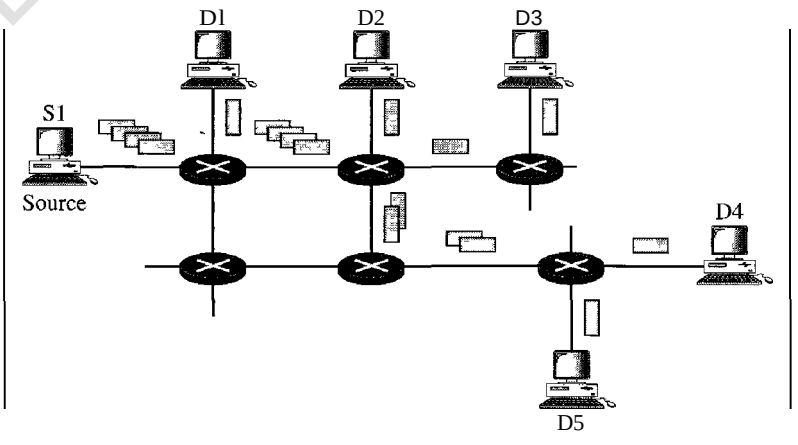
Multicasting Versus Multiple Unicasting

Before we finish this section, we need to distinguish between multicasting and multiple unicasting. Figure 22.35 illustrates both concepts.

Figure 22.35 *Multicasting versus multiple unicasting*



a. Multicasting



b. Multiple nnicasting

Multicasting starts with one single packet from the source that is duplicated by the routers. The destination address in each packet is the same for all duplicates. Note that only one single copy of the packet travels between any two routers.

In multiple unicasting, several packets start from the source. If there are five destinations, for example, the source sends five packets, each with a different unicast destination address. Note that there may be multiple copies traveling between two routers. For example, when a person sends an e-mail message to a group of people, this is multiple unicasting. The e-mail software creates replicas of the message, each with a different destination address and sends them one by one. This is not multicasting; it is multiple unicasting.

Emulation of Multicasting with Unicasting

You might wonder why we have a separate mechanism for multicasting, when it can be emulated with unicasting. There are two obvious reasons for this.

1. Multicasting is more efficient than multiple unicasting. In Figure 22.35, we can see how multicasting requires less bandwidth than does multiple unicasting. In multiple unicasting, some of the links must handle several copies.
2. In multiple unicasting, the packets are created by the source with a relative delay between packets. If there are 1000 destinations, the delay between the first and the last packet may be unacceptable. In multicasting, there is no delay because only one packet is created by the source.

Emulation of multicasting through multiple unicasting is not efficient and may create long delays, particularly with a large group.

Applications

Multicasting has many applications today such as access to distributed databases, information dissemination, teleconferencing, and distance learning.

Access to Distributed Databases

Most of the large databases today are distributed. That is, the information is stored in more than one location, usually at the time of production. The user who needs to access the database does not know the location of the information. A user's request is multicast to all the database locations, and the location that has the information responds.

Information Dissemination

Businesses often need to send information to their customers. If the nature of the information is the same for each customer, it can be multicast. In this way a business can send one message that can reach many customers. For example, a software update can be sent to all purchasers of a particular software package.

Dissemination of News

In a similar manner news can be easily disseminated through multicasting. One single message can be sent to those interested in a particular topic. For example, the statistics of the championship high school basketball tournament can be sent to the sports editors of many newspapers.

Teleconferencing

Teleconferencing involves multicasting. The individuals attending a teleconference all need to receive the same information at the same time. Temporary or permanent groups can be formed for this purpose. For example, an engineering group that holds meetings every Monday morning could have a permanent group while the group that plans the holiday party could form a temporary group.

Distance Learning

One growing area in the use of multicasting is distance learning. Lessons taught by one single professor can be received by a specific group of students. This is especially convenient for those students who find it difficult to attend classes on campus.

Multicast Routing

In this section, we first discuss the idea of optimal routing, common in all multicast protocols. We then give an overview of multicast routing protocols.

Optimal Routing: Shortest Path Trees

The process of optimal interdomain routing eventually results in the finding of the *shortest path tree.* The root of the tree is the source, and the leaves are the potential destinations. The path from the root to each destination is the shortest path. However, the number of trees and the formation of the trees in unicast and multicast routing are different. Let us discuss each separately.

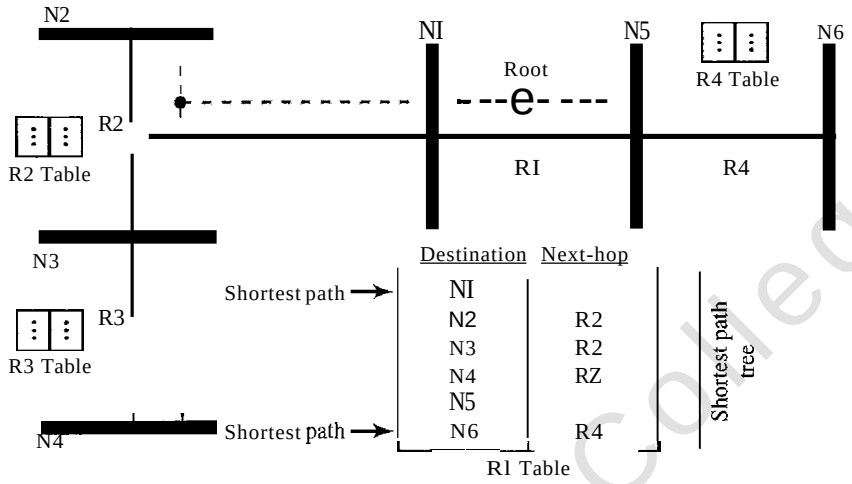
Unicast Routing In unicast routing, when a router receives a packet to forward, it needs to find the shortest path to the destination of the packet. The router consults its routing table for that particular destination. The next-hop entry corresponding to the destination is the start of the shortest path. The router knows the shortest path for each destination, which means that the router has a shortest path tree to optimally reach all destinations. In other words, each line of the routing table is a shortest path; the whole routing table is a shortest path tree. In unicast routing, each router needs only one shortest path tree to forward a packet; however, each router has its own shortest path tree. Figure 22.36 shows the situation.

The figure shows the details of the routing table and the shortest path tree for router R1. Each line in the routing table corresponds to one path from the root to the corresponding network. The whole table represents the shortest path tree.

In unicast routing, each router in the domain has a table that defines a shortest path tree to possible destinations.

Multicast Routing When a router receives a multicast packet, the situation is different from when it receives a unicast packet. A multicast packet may have destinations in more than one network. Forwarding of a single packet to members of a group requires a shortest path tree. If we have *n* groups, we may need *n* shortest path trees. We can imagine the complexity of multicast routing. Two approaches have been used to solve the problem: source-based trees and group-shared trees.

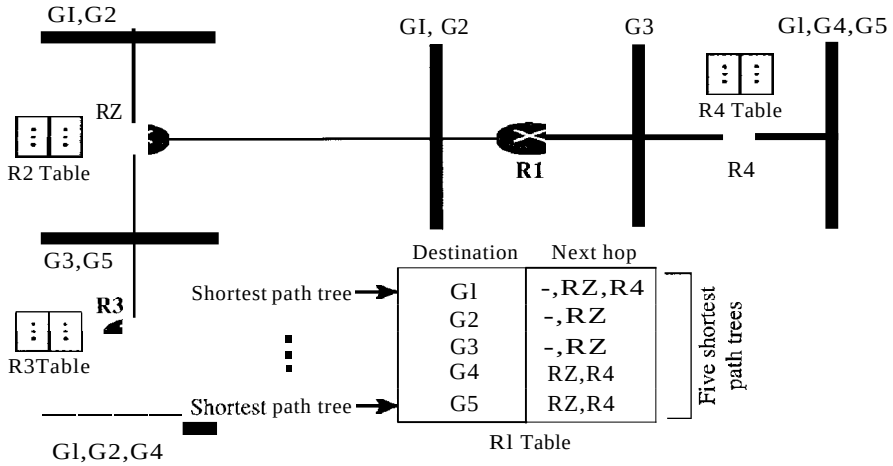
Figure 22.36 *Shortest path tree in unicast routing*



In multicast routing, each involved router needs to construct a shortest path tree for each group.

O Source-Based Tree. In the source-based tree approach, each router needs to have one shortest path tree for each group. The shortest path tree for a group defines the next hop for each network that has loyal member(s) for that group. In Figure 22.37, we assume that we have only five groups in the domain: GI, G2, G3, G4, and G5. At the moment GI has loyal members in four networks, G2 in three, G3 in two, G4 in two, and G5 in two. We have shown the names of the groups with loyal members on each network. Figure 22.37 also shows the multicast routing table for router RI. There is one shortest path tree for each group; therefore there are five shortest path trees for five groups. If router R1 receives a packet with destination

Figure 22.37 *Source-based tree approach*

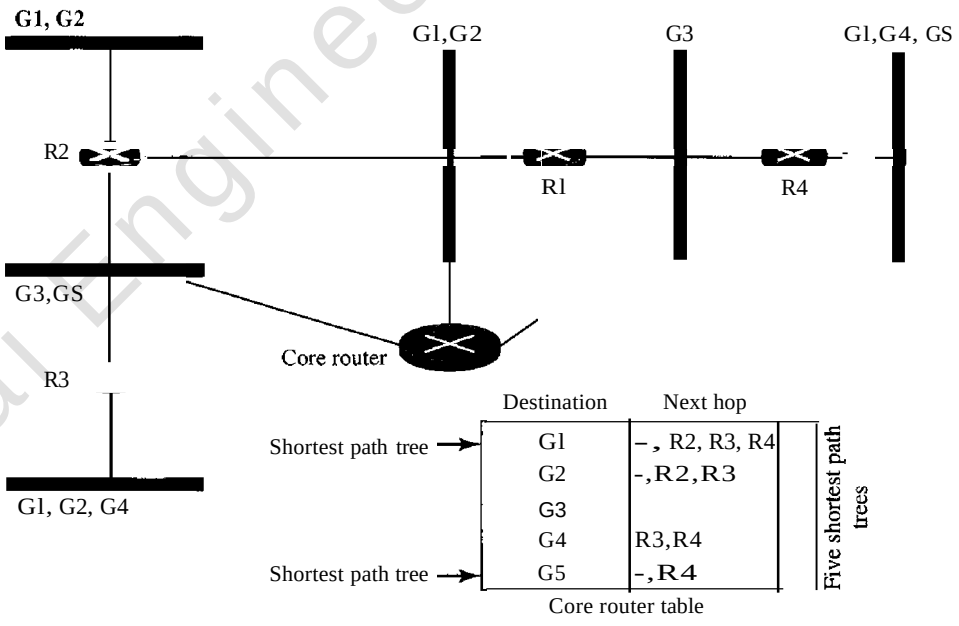


address G1, it needs to send a copy of the packet to the attached network, a copy to router R2, and a copy to router R4 so that all members of G1 can receive a copy. In this approach, if the number of groups is *m,* each router needs to have *m* shortest path trees, one for each group. We can imagine the complexity of the routing table if we have hundreds or thousands of groups. However, we will show how different protocols manage to alleviate the situation.

In the source-based tree approach, each router needs to have one shortest path tree for each group.

- Group-Shared Tree. In the group-shared tree approach, instead of each router having *m* shortest path trees, only one designated router, called the center core, or rendezvous router, takes the responsibility of distributing multicast traffic. The core has *m* shortest path trees in its routing table. The rest of the routers in the domain have none. If a router receives a multicast packet, it encapsulates the packet in a unicast packet and sends it to the core router. The core router removes the multicast packet from its capsule, and consults its routing table to route the packet. Figure 22.38 shows the idea.

Figure 22.38 *Group-shared tree approach*



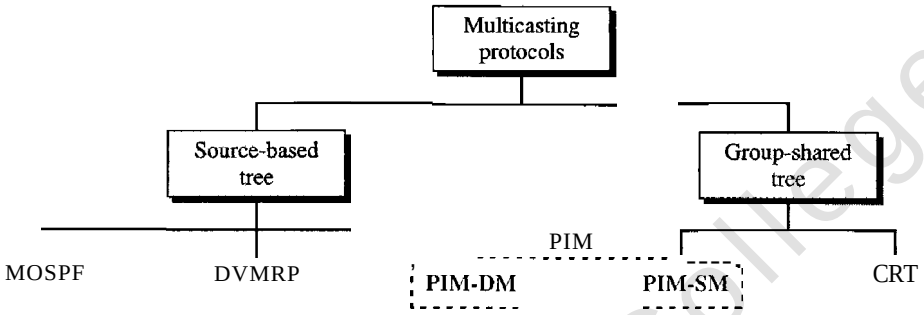
In the group-shared tree approach, only the core router, which has a shortest path tree for each group, is involved in multicasting.

Routing Protocols

During the last few decades, several multicast routing protocols have emerged. Some of these protocols are extensions of unicast routing protocols; others are totally new.

We discuss these protocols in the remainder of this chapter. Figure 22.39 shows the taxonomy of these protocols.

Figure 22.39 *Taxonomy of common multicast protocols*



Multicast Link State Routing: MOSPF

In this section, we briefly discuss multicast link state routing and its implementation in the Internet, MOSPF.

Multicast Link State Routing We discussed unicast link state routing in Section 22.3. We said that each router creates a shortest path tree by using Dijkstra's algorithm. The routing table is a translation of the shortest path tree. Multicast link state routing is a direct extension of unicast routing and uses a source-based tree approach. Although unicast routing is quite involved, the extension to multicast routing is very simple and straightforward.

Multicast link state routing uses the source-based tree approach.

Recall that in unicast routing, each node needs to advertise the state of its links. For multicast routing, a node needs to revise the interpretation of *state.* A node advertises every group which has any loyal member on the link. Here the meaning of state is "what groups are active on this link." The information about the group comes from IGMP (see Chapter 21). Each router running IGMP solicits the hosts on the link to find out the membership status.

When a router receives all these LSPs, it creates *n (n* is the number of groups) topologies, from which *n* shortest path trees are made by using Dijkstra's algorithm. So each router has a routing table that represents as many shortest path trees as there are groups.

The only problem with this protocol is the time and space needed to create and save the many shortest path trees. The solution is to create the trees only when needed. When a router receives a packet with a multicast destination address, it runs the Dijkstra algorithm to calculate the shortest path tree for that group. The result can be cached in case there are additional packets for that destination.

MOSPF Multicast Open Shortest Path First (MOSPF) protocol is an extension of the OSPF protocol that uses multicast link state routing to create source-based trees.

The protocol requires a new link state update packet to associate the unicast address of a host with the group address or addresses the host is sponsoring. This packet is called the group-membership LSA. In this way, we can include in the tree only the hosts (using their unicast addresses) that belong to a particular group. In other words, we make a tree that contains all the hosts belonging to a group, but we use the unicast address of the host in the calculation. For efficiency, the router calculates the shortest path trees on demand (when it receives the first multicast packet). In addition, the tree can be saved in cache memory for future use by the same source/group pair. MOSPF is a data-driven protocol; the first time an MOSPF router sees a datagram with a given source and group address, the router constructs the Dijkstra shortest path tree.

Multicast Distance Vector: DVMRP

In this section, we briefly discuss multicast distance vector routing and its implementation in the Internet, DVMRP.

Multicast Distance Vector Routing Unicast distance vector routing is very simple; extending it to support multicast routing is complicated. Multicast routing does not allow a router to send its routing table to its neighbors. The idea is to create a table from scratch by using the information from the unicast distance vector tables.

Multicast distance vector routing uses source-based trees, but the router never actually makes a routing table. When a router receives a multicast packet, it forwards the packet as though it is consulting a routing table. We can say that the shortest path tree is evanescent. After its use (after a packet is forwarded) the table is destroyed.

To accomplish this, the multicast distance vector algorithm uses a process based on four decision-making strategies. Each strategy is built on its predecessor. We explain them one by one and see how each strategy can improve the shortcomings of the previous one.

- **Flooding.** Flooding is the first strategy that comes to mind. A router receives a packet and, without even looking at the destination group address, sends it out from every interlace except the one from which it was received. Flooding accomplishes the first goal of multicasting: every network with active members receives the packet. However, so will networks without active members. This is a broadcast, not a multicast. There is another problem: it creates loops. A packet that has left the router may come back again from another interlace or the same interlace and be forwarded again. Some flooding protocols keep a copy of the packet for a while and discard any duplicates to avoid loops. The next strategy, reverse path forwarding, corrects this defect.

Flooding broadcasts packets, but creates loops in the systems.

- **Reverse Path Forwarding (RPF).** RPF is a modified flooding strategy. To prevent loops, only one copy is forwarded; the other copies are dropped. In RPF, a router forwards only the copy that has traveled the shortest path from the source to the router. To find this copy, RPF uses the unicast routing table. The router receives a packet and extracts the source address (a unicast address). It consults its unicast routing table as though it wants to send a packet to the source address. The routing table tells the

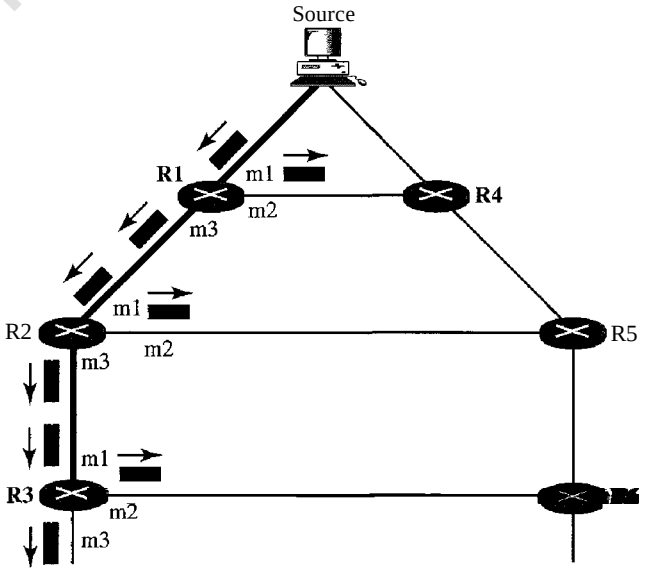
router the next hop. If the multicast packet has just come from the hop defined in the table, the packet has traveled the shortest path from the source to the router because the shortest path is reciprocal in unicast distance vector routing protocols. If the path from A to B is the shortest, then it is also the shortest from B to A. The router forwards the packet if it has traveled from the shortest path; it discards it otherwise.

This strategy prevents loops because there is always one shortest path from the source to the router. If a packet leaves the router and comes back again, it has not traveled the shortest path. To make the point clear, let us look at Figure 22.40.

Figure 22.40 shows part of a domain and a source. The shortest path tree as calculated by routers RI, R2, and R3 is shown by a thick line. When RI receives a packet from the source through the interface rnl, it consults its routing table and finds that the shortest path from RI to the source is through interface mI. The packet is forwarded. However, if a copy of the packet has arrived through interface m2, it is discarded because m2 does not define the shortest path from RI to the source. The story is the same with R2 and R3. You may wonder what happens if a copy of a packet that arrives at the ml interface of R3, travels through R6, R5, R2, and then enters R3 through interface ml. This interface is the correct interface for R3. Is the copy of the packet forwarded? The answer is that this scenario never happens because when the packet goes from R5 to R2, it will be discarded by R2 and never reaches R3. The upstream routers toward the source always discard a packet that has not gone through the shortest path, thus preventing confusion for the downstream routers.

RPF eliminates the loop in the flooding process.

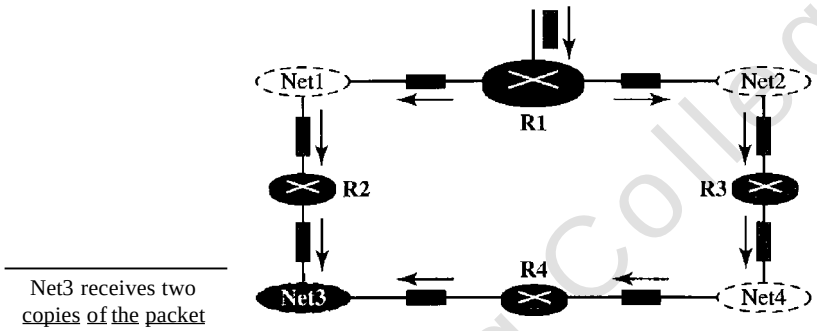
Figure 22.40 *Reverse path forwarding (RPF)*



- Reverse Path Broadcasting (RPB). RPF guarantees that each network receives a copy of the multicast packet without formation of loops. However, RPF does not

guarantee that each network receives only one copy; a network may receive two or more copies. The reason is that RPF is not based on the destination address (a group address); forwarding is based on the source address. To visualize the problem, let us look at Figure 22.41.

Figure 22.41 *Problem with RPF*

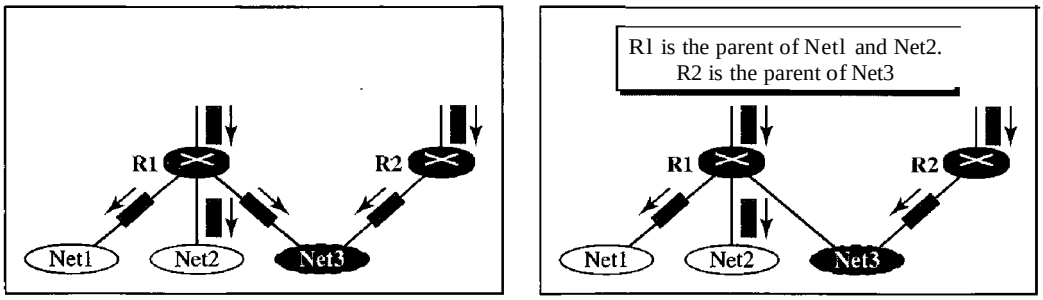


Net3 in this figure receives two copies of the packet even though each router just sends out one copy from each interface. There is duplication because a tree has not been made; instead of a tree we have a graph. Net3 has two parents: routers R2 andR4.

To eliminate duplication, we must define only one parent router for each network. We must have this restriction: A network can receive a multicast packet from a particular source only through a designated parent router.

Now the policy is clear. For each source, the router sends the packet only out of those interfaces for which it is the designated parent. This policy is called reverse path broadcasting (RPB). RPB guarantees that the packet reaches every network and that every network receives only one copy. Figure 22.42 shows the difference between RPF and RPB.

Figure 22.42 *RPF Versus RPB*



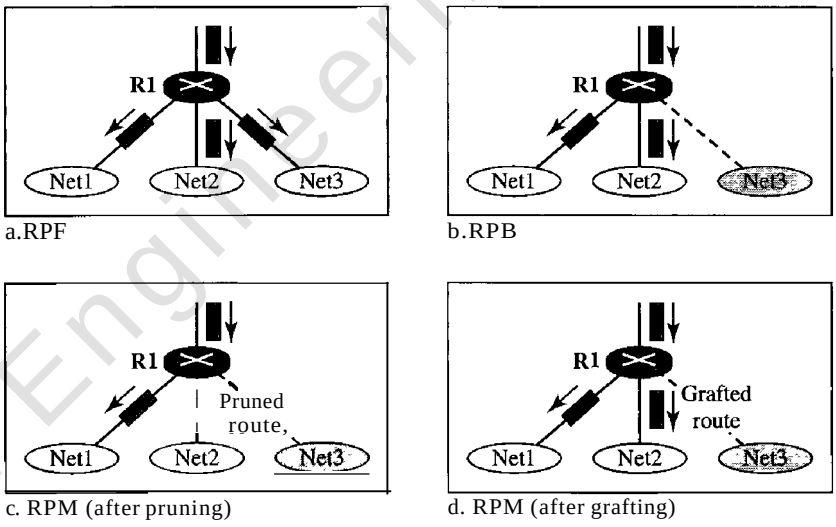
The reader may ask how the designated parent is determined. The designated parent router can be the router with the shortest path to the source. Because routers periodically

send updating packets to each other (in RIP), they can easily determine which router in the neighborhood has the shortest path to the source (when interpreting the source as the destination), If more than one router qualifies, the router with the smallest IP address is selected.

RPB creates a shortest path broadcast tree from the source to each destination. It guarantees that each destination receives one and only one copy of the packet.

- Reverse Path Multicasting (RPM). As you may have noticed, RPB does not multicast the packet, it broadcasts it. This is not efficient. To increase efficiency, the multicast packet must reach only those networks that have active members for that particular group. This is called reverse path multicasting (RPM). To convert broadcasting to multicasting, the protocol uses two procedures, pruning and grafting. Figure 22.43 shows the idea of pruning and grafting.

Figure 22.43 *RPF, RPB, and RPM*



The designated parent router of each network is responsible for holding the membership information. This is done through the IGMP protocol described in Chapter 21. The process starts when a router connected to a network finds that there is no interest in a multicast packet. The router sends a prune message to the upstream router so that it can exclude the corresponding interface. That is, the upstream router can stop sending multicast messages for this group through that interface. Now if this router receives prune messages from all downstream routers, it, in turn, sends a prune message to its upstream router.

What if a leaf router (a router at the bottom of the tree) has sent a prune message but suddenly realizes, through IGMP, that one of its networks is again interested in receiving the multicast packet? It can send a graft message. The graft message forces the upstream router to resume sending the multicast messages.

RPM adds pruning and grafting to RPB to create a multicast shortest path tree that supports dynamic membership changes.

DVMRP The Distance Vector Multicast Routing Protocol (DVMRP) is an implementation of multicast distance vector routing. It is a source-based routing protocol, based on RIP.

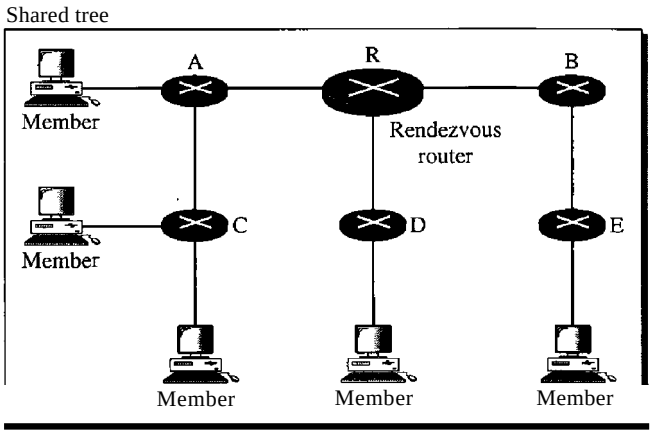
CRT

The Core-Based Tree (CBT) protocol is a group-shared protocol that uses a core as the root of the tree. The autonomous system is divided into regions, and a core (center router or rendezvous router) is chosen for each region.

Formation of the Tree After the rendezvous point is selected, every router is infonned of the unicast address of the selected router. Each router then sends a unicast join message (similar to a grafting message) to show that it wants to join the group. This message passes through all routers that are located between the sender and the rendezvous router. Each intermediate router extracts the necessary infonnation from the message, such as the unicast address of the sender and the interface through which the packet has arrived, and forwards the message to the next router in the path. When the rendezvous router has received all join messages from every member of the group, the tree is formed. Now every router knows its upstream router (the router that leads to the root) and the downstream router (the router that leads to the leaf).

If a router wants to leave the group, it sends a leave message to its upstream router. The upstream router removes the link to that router from the tree and forwards the message to its upstream router, and so on. Figure 22.44 shows a group-shared tree with its rendezvous router.

Figure 22.44 *Group-shared tree with rendezvous router*

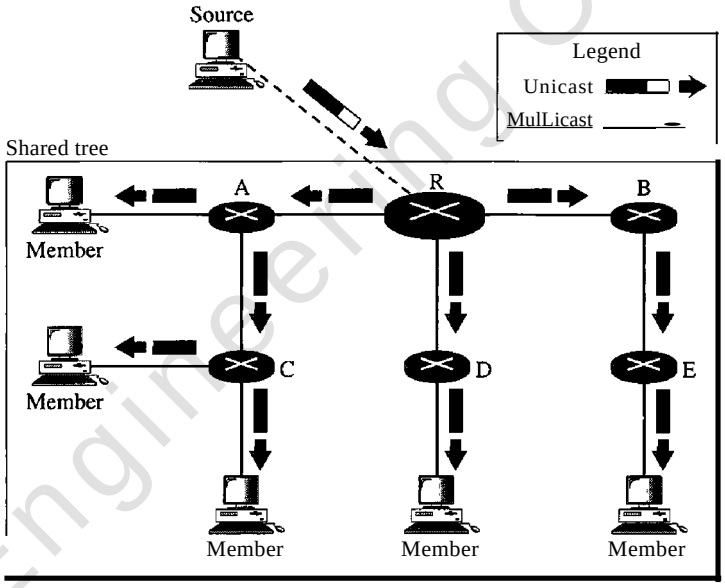


The reader may have noticed two differences between DVMRP and MOSPF, on one hand, and CBT, on the other. First, the tree for the first two is made from the root up; the tree for CBT is fonned from the leaves down. Second, in DVMRP, the tree is

first made (broadcasting) and then pruned; in CBT, there is no tree at the beginning; the joining (grafting) gradually makes the tree.

Sending Multicast Packets After formation of the tree, any source (belonging to the group or not) can send a multicast packet to all members of the group. It simply sends the packet to the rendezvous router, using the unicast address of the rendezvous router; the rendezvous router distributes the packet to all members of the group. Figure 22.45 shows how a host can send a multicast packet to all members of the group. Note that the source host can be any of the hosts inside the shared tree or any host outside the shared tree. In Figure 22.45 we show one located outside the shared tree.

Figure 22.45 *Sending a multicast packet to the rendezvous router*



Selecting the Rendezvous Router This approach is simple except for one point. How do we select a rendezvous router to optimize the process and multicasting as well? Several methods have been implemented. However, this topic is beyond the scope of this book, and we leave it to more advanced books.

In summary, the Core-Based Tree (CBT) is a group-shared tree, center-based protocol using one tree per group. One of the routers in the tree is called the core. A packet is sent from the source to members of the group following this procedure:

1. The source, which mayor may not be part of the tree, encapsulates the multicast packet inside a unicast packet with the unicast destination address of the core and sends it to the core. This part of delivery is done using a unicast address; the only recipient is the core router.
2. The core decapsulates the unicast packet and forwards it to all interested intetfaces.
3. Each router that receives the multicast packet, in tum, forwards it to all interested interfaces.

In CRT, the source sends the multicast packet (encapsulated in a unicast packet) to the core router. The core router decapsulates the packet and forwards it to all interested interfaces.

PIM

Protocol Independent Multicast (PIM) is the name given to two independent multicast routing protocols: Protocol Independent Multicast, Dense Mode (PIM-DM) and Protocol Independent Multicast, Sparse Mode (PIM-SM). Both protocols are unicast-protocol-dependent, but the similarity ends here. We discuss each separately.

PIM-DM PIM-DM is used when there is a possibility that each router is involved in multicasting (dense mode). In this environment, the use of a protocol that broadcasts the packet is justified because almost all routers are involved in the process.

PIM-DM is used in a dense multicast environment, such as a LAN.

PIM-DM is a source-based tree routing protocol that uses RPF and pruning and grafting strategies for multicasting. Its operation is like that of DVMRP; however, unlike DVMRP, it does not depend on a specific unicasting protocol. It assumes that the autonomous system is using a unicast protocol and each router has a table that can find the outgoing interface that has an optimal path to a destination. This unicast protocol can be a distance vector protocol (RIP) or link state protocol (OSPF).

PIM-DM uses RPF and pruning and grafting strategies to handle multicasting. However, it is independent of the underlying unicast protocol.

PIM-SM PIM-SM is used when there is a slight possibility that each router is involved in multicasting (sparse mode). In this environment, the use of a protocol that broadcasts the packet is not justified; a protocol such as CBT that uses a group-shared tree is more appropriate.

PIM-SM is used in a sparse multicast environment such as a WAN.

PIM-SM is a group-shared tree routing protocol that has a rendezvous point (RP) as the source of the tree. Its operation is like CBT; however, it is simpler because it does not require acknowledgment from a join message. In addition, it creates a backup set of RPs for each region to cover RP failures.

One of the characteristics of PIM-SM is that it can switch from a group-shared tree strategy to a source-based tree strategy when necessary. This can happen if there is a dense area of activity far from the RP. That area can be more efficiently handled with a source-based tree strategy instead of a group-shared tree strategy.

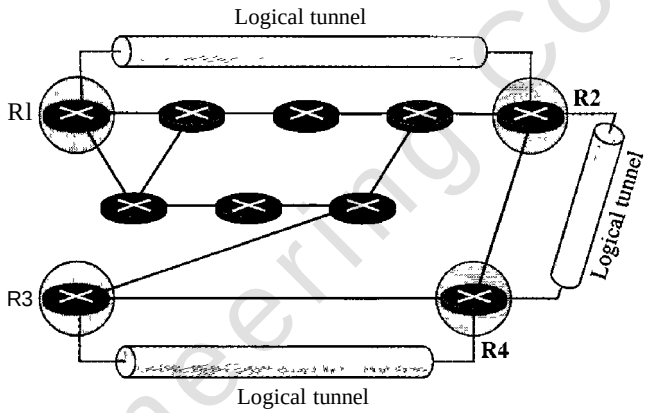
PIM-SM is similar to CRT but uses a simpler procedure.

MBONE

Multimedia and real-time communication have increased the need for multicasting in the Internet. However, only a small fraction of Internet routers are multicast routers. In

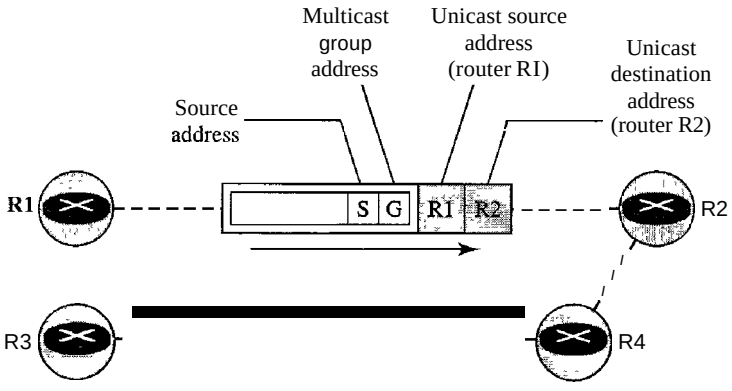
other words, a multicast router may not find another multicast router in the neighborhood to forward the multicast packet. Although this problem may be solved in the next few years by adding more and more multicast routers, there is another solution to this problem. The solution is tunneling. The multicast routers are seen as a group of routers on top of unicast routers. The multicast routers may not be connected directly, but they are connected logically. Figure 22.46 shows the idea. In Figure 22.46, only the routers enclosed in the shaded circles are capable of multicasting. Without tunneling, these routers are isolated islands. To enable multicasting, we make a multicast backbone (MBONE) out of these isolated routers by using the concept of tunneling.

Figure 22.46 *Logical tunneling*



A logical tunnel is established by encapsulating the multicast packet inside a unicast packet. The multicast packet becomes the payload (data) of the unicast packet. The intermediate (nonmulticast) routers forward the packet as unicast routers and deliver the packet from one island to another. It's as if the unicast routers do not exist and the two multicast routers are neighbors. Figure 22.47 shows the concept. So far the only protocol that supports MBONE and tunneling is DVMRP.

Figure 22.47 *MBONE*



22.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

Delivery and forwarding are discussed in Chapter 6 of [For06]. Unicast routing protocols are discussed in Chapter 14 of [For06]. Multicasting and multicast routing are discussed in Chapter 15 of [For06]. For a complete discussion of multicasting see [WZOl]. For routing protocols see [HuiOO]. OSPF is discussed in [Moy98].

Sites

- www.ietf.org/rfc.html Information about RFCs

RFCs

A discussion of RIP can be found in the following RFCs:

1131,1245,1246,1247,1370,1583,1584,1585, 1586,1587, 1722,1723,2082,2453

A discussion of OSPF can be found in the following RFCs:

1131,1245,1246,1247,1370,1583,1584,1585, 1586, 1587,2178,2328,2329,2370

A discussion of BGP can be found in the following RFCs:

1092,1105,1163,1265,1266,1267,1364,1392,1403, 1565,1654,1655,1665,1771,1772, 1745,1774,2283

22.6 KEY TERMS

address aggregation
area
area border routers
area identification
autonomous system (AS)
backbone router
Border Gateway Protocol (BGP)
broadcasting
Core-Based Tree (CBT) protocol
data-driven
default method
delivery
designated parent router
Dijkstra's algorithm
direct delivery
distance learning
Distance Vector Multicast Routing Protocol (DVMRP)
distance vector routing
distributed database
dynamic routing method
dynamic routing table
flooding
forwarding
graft message
group-shared tree

hierarchical routing
hop count
host-specific method
ifconfig
immediate neighbors
indirect delivery
interdomain routing
intradomain routing
least-cost tree
link state routing
logical tunnel
longest mask matching
metric
multicast backbone (MBONE)
Multicast Open Shortest Path First (MOSPF)
multicast router
multicast routing
multicasting
multiple unicasting
netstat
network-specific method
next-hop address
next-hop method
Open Shortest Path First (OSPF)
optional attribute
OSPF protocol
path vector routing
point-to-point link
poison reverse
policy routing
Protocol Independent Multicast (PIM)
Protocol Independent Multicast, Dense Mode (PIM-DM)
Protocol Independent Multicast, Sparse Mode (PIM-SM)
prune message
rendezvous router
rendezvous-point tree
reverse path broadcasting (RPB)
reverse path forwarding (RPF)
reverse path multicasting (RPM)
route method
routing
Routing Information Protocol (RIP)
routing protocols
shortest path tree
slow convergence
source-based tree
speaker node
split horizon
static routing table
stub link
switching fabric
teleconferencing
transient link
triggered update
tunneling
unicasting
update message
virtual link
well-known attribute

22.7 SUMMARY

- The delivery of a packet is called direct if the deliverer (host or router) and the destination are on the same network; the delivery of a packet is called indirect if the deliverer (host or router) and the destination are on different networks.
- In the next-hop method, instead of a complete list of the stops the packet must make, only the address of the next hop is listed in the routing table; in the network-specific method, all hosts on a network share one entry in the routing table.

- In the host-specific method, the full IP address of a host is given in the routing table.
- In the default method, a router is assigned to receive all packets with no match in the routing table.
- The routing table for classless addressing needs at least four columns.
- Address aggregation simplifies the forwarding process in classless addressing.
- Longest mask matching is required in classless addressing.
- Classless addressing requires hierarchical and geographical routing to prevent immense routing tables.
- A static routing table's entries are updated manually by an administrator; a dynamic routing table's entries are updated automatically by a routing protocol.
- A metric is the cost assigned for passage of a packet through a network.
- An autonomous system (AS) is a group of networks and routers under the authority of a single administration.
- RIP is based on distance vector routing, in which each router shares, at regular intervals, its knowledge about the entire AS with its neighbors.
- Two shortcomings associated with the RIP protocol are slow convergence and instability. Procedures to remedy RIP instability include triggered update, split horizons, and poison reverse.
- OSPF divides an AS into areas, defined as collections of networks, hosts, and routers.
- OSPF is based on link state routing, in which each router sends the state of its neighborhood to every other router in the area. A packet is sent only if there is a change in the neighborhood.
- OSPF routing tables are calculated by using Dijkstra's algorithm.
- BGP is an interautonomous system routing protocol used to update routing tables.
- BGP is based on a routing protocol called path vector routing. In this protocol, the ASs through which a packet must pass are explicitly listed.
- In a source-based tree approach to multicast routing, the source/group combination determines the tree; in a group-shared tree approach to multicast routing, the group determines the tree.
- MOSPF is a multicast routing protocol that uses multicast link state routing to create a source-based least-cost tree.
- In reverse path forwarding (RPF), the router forwards only the packets that have traveled the shortest path from the source to the router.
- Reverse path broadcasting (RPB) creates a shortest path broadcast tree from the source to each destination. It guarantees that each destination receives one and only one copy of the packet.
- Reverse path multicasting (RPM) adds pruning and grafting to RPB to create a multicast shortest path tree that supports dynamic membership changes.
- DVMRP is a multicast routing protocol that uses the distance routing protocol to create a source-based tree.
- The Core-Based Tree (CBT) protocol is a multicast routing protocol that uses a router as the root of the tree.

D PIM-DM is a source-based tree routing protocol that uses RPF and pruning and grafting strategies to handle multicasting.

O PIM-SM is a group-shared tree routing protocol that is similar to CBT and uses a rendezvous router as the source of the tree.

D For multicasting between two noncontiguous multicast routers, we make a multicast backbone (MBONE) to enable tunneling.

22.8 PRACTICE SET

Review Questions

1. What is the difference between a direct and an indirect delivery?
2. List three forwarding techniques discussed in this chapter and give a brief description of each.
3. Contrast two different routing tables discussed in this chapter.
4. What is the purpose of RIP?
5. What are the functions of a RIP message?
6. Why is the expiration timer value 6 times that of the periodic timer value?
7. How does the hop count limit alleviate RIP's problems?
8. List RIP shortcomings and their corresponding fixes.
9. What is the basis of classification for the four types of links defined by OSPF?
10. Why do OSPF messages propagate faster than RIP messages?
11. What is the purpose of BGP?
12. Give a brief description of two groups of multicast routing protocols discussed in this chapter.

Exercises

13. Show a routing table for a host that is totally isolated.
14. Show a routing table for a host that is connected to a LAN without being connected to the Internet.
15. Find the topology of the network if Table 22.3 is the routing table for router R1.

Table 22.3 *Routing table for Exercise 15*

Mask	*Network Address*	*Next-Hop Address*	*Interface*
/27	202.14.17.224	-	m1
/18	145.23.192.0	-	mO
Default	Default	130.56.12.4	m2

16. Can router R1 in Figure 22.8 receive a packet with destination address 140.24.7.194? Explain your answer.

17. Can router Rl in Figure 22.8 receive a packet with destination address l40.24.7.42? Explain your answer.
18. Show the routing table for the regional ISP in Figure 22.9.
19. Show the routing table for local ISP 1 in Figure 22.9.
20. Show the routing table for local ISP 2 in Figure 22.9.
21. Show the routing table for local ISP 3 in Figure 22.9.
22. Show the routing table for small ISP 1 in Figure 22.9.
23. Contrast and compare distance vector routing with link state routing.
24. A router has the following RIP routing table:

Net!	4	B
Net2	2	C
Net3	!	F
Net4	5	G

What would be the contents of the table if the router received the following RIP message from router C?

Net!	2
Net2	!
Net3	3
Net4	7

25. How many bytes are empty in a RIP message that advertises *N* networks?
26. A router has the following RIP routing table:

Net!	4	B
Net2	2	C
Net3	1	F
Net4	5	G

Show the response message sent by this router.

27. Show the autonomous system with the following specifications:
    a. There are eight networks (Nl to N8).
    b. There are eight routers (Rl to R8).
    c. Nl, N2, N3, N4, NS, and N6 are Ethernet LANs.
    d. N7 and N8 are point-to-point WANs.
    e. Rl connects Nl and N2.
    f. R2 connects Nl and N7.
    g. R3 connects N2 and N8.
    h. R4 connects N7 and N6.
    i. RS connects N6 and N3.
    j. R6 connects N6 and N4.
    k. R7 connects N6 and NS.
    1. R8 connects N8 and N5.

28. Draw the graphical representation of the autonomous system of Exercise 27 as seen by OSPF.
29. Which of the networks in Exercise 27 is a transient network? Which is a stub network?
30. A router using DVMRP receives a packet with source address 10.14.17.2 from interface 2. If the router forwards the packet, what are the contents of the entry related to this address in the unicast routing table?
31. Does RPF actually create a shortest path tree? Explain.
32. Does RPB actually create a shortest path tree? Explain. What are the leaves of the tree?
33. Does RPM actually create a shortest path tree? Explain. What are the leaves of the tree?

Research Activities

34. If you have access to UNIX (or LINUX), use *netstat* and *ifconfig* to find the routing table for the server to which you are connected.
35. Find out how your ISP uses address aggregation and longest mask match principles.
36. Find out whether your IP address is part of the geographical address allocation.
37. If you are using a router, find the number and names of the columns in the routing table.

PART 5

Transport Layer

Objectives

The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on a host. Whereas the network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.

The transport layer is responsible for the delivery of a message from one process to another.

Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process on one computer to a specific process on the other. The transport layer header must therefore include a type of address called a *service-point address* in the OSI model and port number or port addresses in the Internet and TCP/IP protocol suite.

A transport layer protocol can be either connectionless or connection-oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data is transferred, the connection is terminated.

In the transport layer, a message is normally divided into transmittable segments. A connectionless protocol, such as UDP, treats each segment separately. A connection-oriented protocol, such as TCP and SCTP, creates a relationship between the segments using sequence numbers.

Like the data link layer, the transport layer may be responsible for flow and error control. However, flow and error control at this layer is performed end to end rather than across a single link. We will see that one of the protocols discussed in this part of

the book, UDP, is not involved in flow or error controL On the other hand, the other two protocols, TCP and SCTP, use sliding windows for flow control and an acknowledgment system for error controL

Part 5 of the book is devoted to the transport layer
and the services provided by this layer.

Chapters

This part consists of two chapters: Chapters 23 and 24.

Chapter 23

Chapter 23 discusses three transport layer protocols in the Internet: UDP, TCP, and SCTP. The first, User Datagram Protocol (UDP), is a connectionless, unreliable protocol that is used for its efficiency. The second, Transmission Control Protocol (TCP), is a connection-oriented, reliable protocol that is a good choice for data transfer. The third, Stream Control Transport Protocol (SCTP) is a new transport-layer protocol designed for multimedia applications.

Chapter 24

Chapter 24 discuss two related topics: congestion control and quality of service. Although these two issues can be related to any layer, we discuss them here with some references to other layers.

CHAPTER 23

Process-la-Process Delivery: UDp, TCp, and SCTP

We begin this chapter by giving the rationale for the existence of the transport layer-the need for process-to-process delivery. We discuss the issues arising from this type of delivery, and we discuss methods to handle them.

The Internet model has three protocols at the transport layer: UDP, TCP, and SCTP. First we discuss UDP, which is the simplest of the three. We see how we can use this very simple transport layer protocol that lacks some of the features of the other two.

We then discuss TCP, a complex transport layer protocol. We see how our previously presented concepts are applied to TCP. We postpone the discussion of congestion control and quality of service in TCP until Chapter 24 because these two topics apply to the data link layer and network layer as well.

We finally discuss SCTP, the new transport layer protocol that is designed for multihomed, multistream applications such as multimedia.

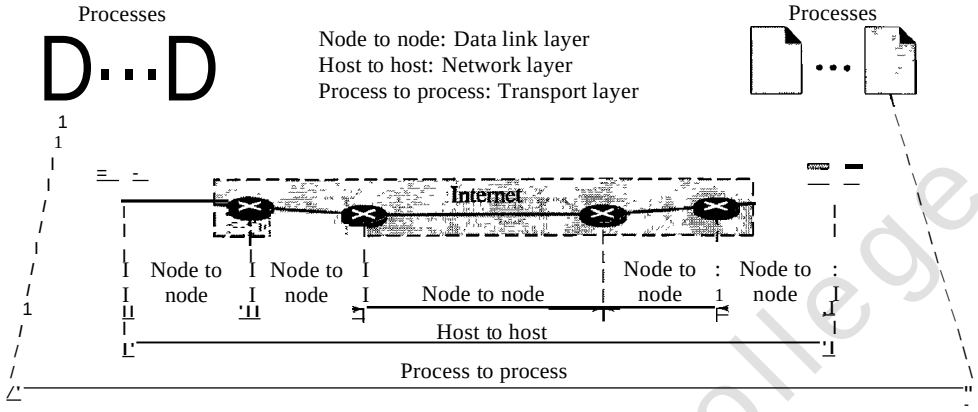
23.1 PROCESS-TO-PROCESS DELIVERY

The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called *node-to-node delivery.* The network layer is responsible for delivery of datagrams between two hosts. This is called *host-to-host delivery.* Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts. Real communication takes place between two processes (application programs). We need process-to-process delivery. However, at any moment, several processes may be running on the source host and several on the destination host. To complete the delivery, we need a mechanism to deliver data from one of these processes running on the source host to the corresponding process running on the destination host.

The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later. Figure 23.1 shows these three types of deliveries and their domains.

The transport layer is responsible for process-to-process delivery.

Figure 23.1 *Types of data deliveries*



Client/Server Paradigm

Although there are several ways to achieve process-to-process communication, the most common one is through the client/server paradigm. A process on the local host, called a client, needs services from a process usually on the remote host, called a server.

Both processes (client and server) have the same name. For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

Operating systems today support both multiuser and multiprogramming environments. A remote computer can run several server programs at the same time, just as local computers can run one or more client programs at the same time. For communication, we must define the following:

1. Local host
2. Local process
3. Remote host
4. Remote process

Addressing

Whenever we need to deliver something to one specific destination among many, we need an address. At the data link layer, we need a MAC address to choose one node among several nodes if the connection is not point-to-point. A frame in the data link layer needs a destination MAC address for delivery and a source address for the next node's reply.

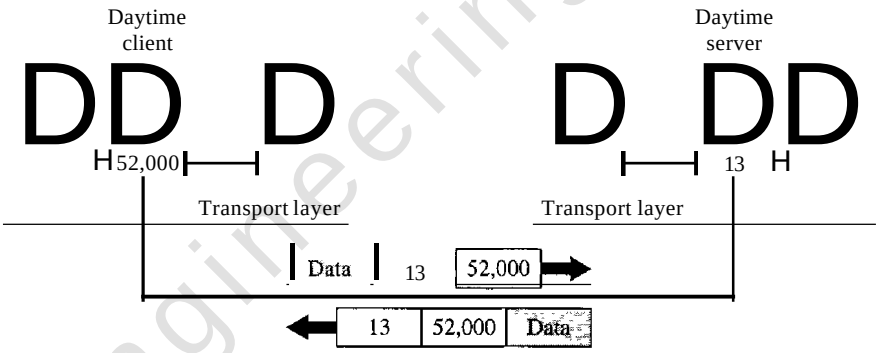
At the network layer, we need an IP address to choose one host among millions. A datagram in the network layer needs a destination IP address for delivery and a source IP address for the destination's reply.

At the transport layer, we need a transport layer address, called a port number, to choose among multiple processes running on the destination host. The destination port number is needed for delivery; the source port number is needed for the reply.

In the Internet model, the port numbers are 16-bit integers between 0 and 65,535. The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the ephemeral port number.

The server process must also define itself with a port number. This port number, however, cannot be chosen randomly. If the computer at the server site runs a server process and assigns a random number as the port number, the process at the client site that wants to access that server and use its services will not know the port number. Of course, one solution would be to send a special packet and request the port number of a specific server, but this requires more overhead. The Internet has decided to use universal port numbers for servers; these are called well-known port numbers. There are some exceptions to this rule; for example, there are clients that are assigned well-known port numbers. Every client process knows the well-known port number of the corresponding server process. For example, while the Daytime client process, discussed above, can use an ephemeral (temporary) port number 52,000 to identify itself, the Daytime server process must use the well-known (permanent) port number 13. Figure 23.2 shows this concept.

Figure 23.2 *Port numbers*



It should be clear by now that the IP addresses and port numbers play different roles in selecting the final destination of data. The destination IP address defines the host among the different hosts in the world. After the host has been selected, the port number defines one of the processes on this particular host (see Figure 23.3).

lANA Ranges

The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private), as shown in Figure 23.4.

- Well-known ports. The ports ranging from 0 to 1023 are assigned and controlled by lANA. These are the well-known ports.
- Registered ports. The ports ranging from 1024 to 49,151 are not assigned or controlled by lANA. They can only be registered with lANA to prevent duplication.
- Dynamic ports. The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral ports.

Figure 23.3 *IP addresses versus port numbers*

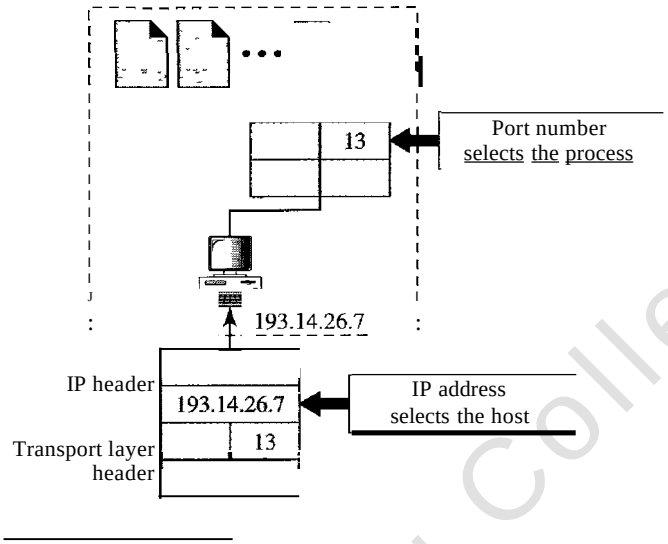
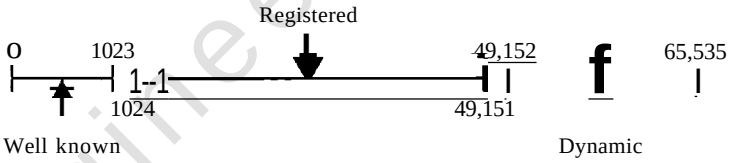


Figure 23.4 *lANA ranges*

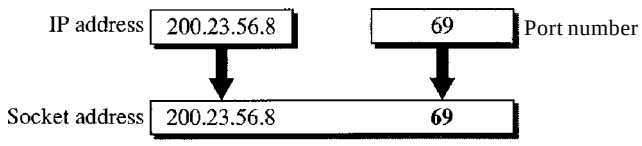


Socket Addresses

Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely (see Figure 23.5).

A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address. These four pieces of information are part of the IP header and the transport layer protocol header. The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.

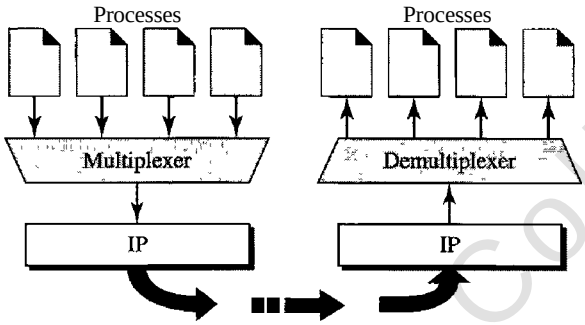
Figure 23.5 *Socket address*



Multiplexing and Demultiplexing

The addressing mechanism allows multiplexing and demultiplexing by the transport layer, as shown in Figure 23.6.

Figure 23.6 *Multiplexing and demultiplexing*



Multiplexing

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing. The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing

At the receiver site, the relationship is one-to-many and requires demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

Connectionless Versus Connection-Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

Connectionless Service

In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. We will see shortly that one of the transport layer protocols in the Internet model, UDP, is connectionless.

Connection-Oriented Service

In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released. We will see shortly that TCP and SCTP are connection-oriented protocols.

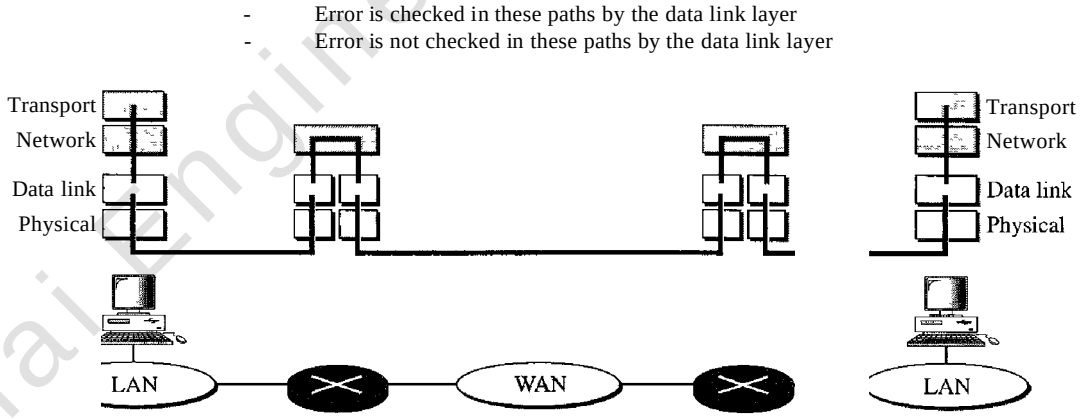
Reliable Versus Unreliable

The transport layer service can be reliable or unreliable. If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service. On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.

In the Internet, there are three common different transport layer protocols, as we have already mentioned. UDP is connectionless and unreliable; TCP and SCTP are connection-oriented and reliable. These three can respond to the demands of the application layer programs.

One question often comes to the mind. If the data link layer is reliable and has flow and error control, do we need this at the transport layer, too? The answer is yes. Reliability at the data link layer is between two nodes; we need reliability between two ends. Because the network layer in the Internet is unreliable (best-effort delivery), we need to implement reliability at the transport layer. To understand that error control at the data link layer does not guarantee error control at the transport layer, let us look at Figure 23.7.

Figure 23.7 *Error control*

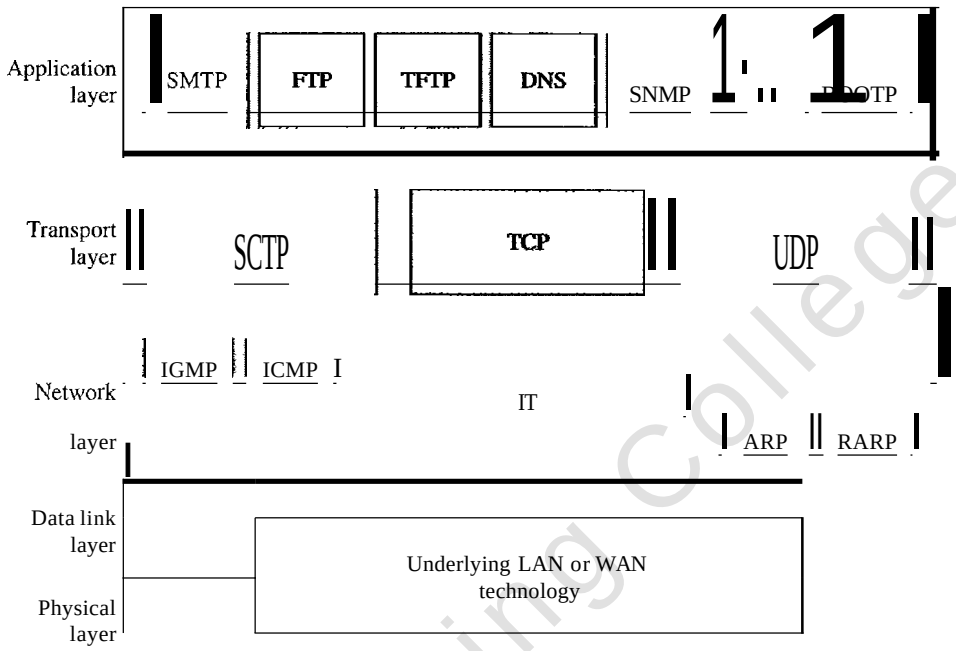


As we will see, flow and error control in TCP is implemented by the sliding window protocol, as discussed in Chapter 11. The window, however, is character-oriented, instead of frame-oriented.

Three Protocols

The original TCP/IP protocol suite specifies two protocols for the transport layer: UDP and TCP. We first focus on UDP, the simpler of the two, before discussing TCP. A new transport layer protocol, SCTP, has been designed, which we also discuss in this chapter. Figure 23.8 shows the position of these protocols in the TCP/IP protocol suite.

Figure 23.8 *Position of UDp, TCp, and SCTP in TCPIIP suite*



23.2 USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs very limited error checking.

If UDP is so powerless, why would a process want to use it? With the disadvantages come some advantages. UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

Well-Known Ports for UDP

Table 23.1 shows some well-known port numbers used by UDP. Some port numbers can be used by both UDP and TCP. We discuss them when we talk about TCP later in the chapter.

Table 23.1 *Well-known ports used with UDP*

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users

Table 23.1 *Well-known ports used with UDP (continued)*

Port	*Protocol*	*Description*
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
III	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Example 23.1

In UNIX, the well-known ports are stored in a file called fetcfservices. Each line in this file gives the name of the server and the well-known port number. We can use the *grep* utility to extract the line corresponding to the desired application. The following shows the port for FTP. Note that FTP can use port 21 with either UDP or TCP.

```
$grep    ftp    fetclservices
ftp          21ftcp
fip          211udp
```

SNMP uses two port numbers (161 and 162), each for a different purpose, as we will see in Chapter 28.

```
$grep    snmp fetclservices
snmp          161ftcp          #Simple Net  Mgmt Proto
snmp          1611udp          #Simple Net  Mgmt Proto
snmptrap      162/udp          #Traps for SNMP
```

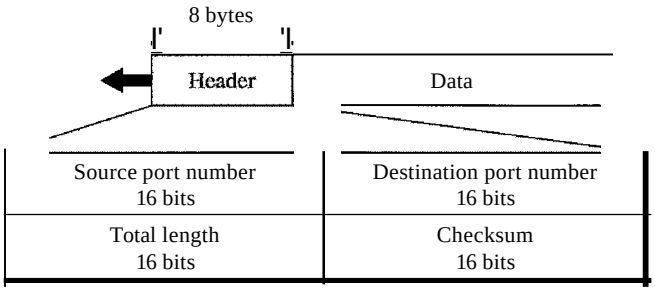
User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. Figure 23.9 shows the format of a user datagram.

The fields are as follows:

- Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Figure 23.9 *User datagram format*



- Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.
- Length. This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.

  The length field in a UDP user datagram is actually not necessary. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length. There is another field in the IP datagram that defines the length of the header. So if we subtract the value of the second field from the first, we can deduce the length of a UDP datagram that is encapsulated in an IP datagram.

  UDP length = IP length - IP header's length

  However, the designers of the UDP protocol felt that it was more efficient for the destination UDP to calculate the length of the data from the information provided in the UDP user datagram rather than ask the IP software to supply this information. We should remember that when the IP software delivers the UDP user datagram to the UDP layer, it has already dropped the IP header.
- Checksum. This field is used to detect errors over the entire user datagram (header plus data). The checksum is discussed next.

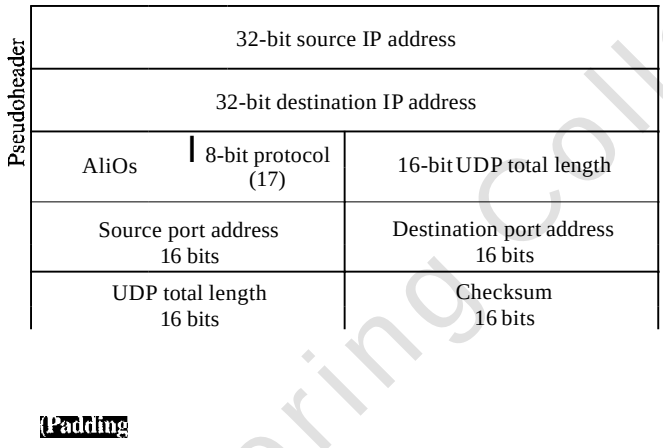
Checksum

We have already talked about the concept of the checksum and the way it is calculated in Chapter 10. We have also shown how to calculate the checksum for the IP and ICMP packet. We now show how this is done for UDP.

The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.

The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with Os (see Figure 23.10).

Figure 23.10 *Pseudoheader for checksum calculation*



If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.

The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols. We will see later that if a process can use either UDP or TCP, the destination port number can be the same. The value of the protocol field for UDP is 17. If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet. It is not delivered to the wrong protocol.

Note the similarities between the pseudoheader fields and the last 12 bytes of the IP header.

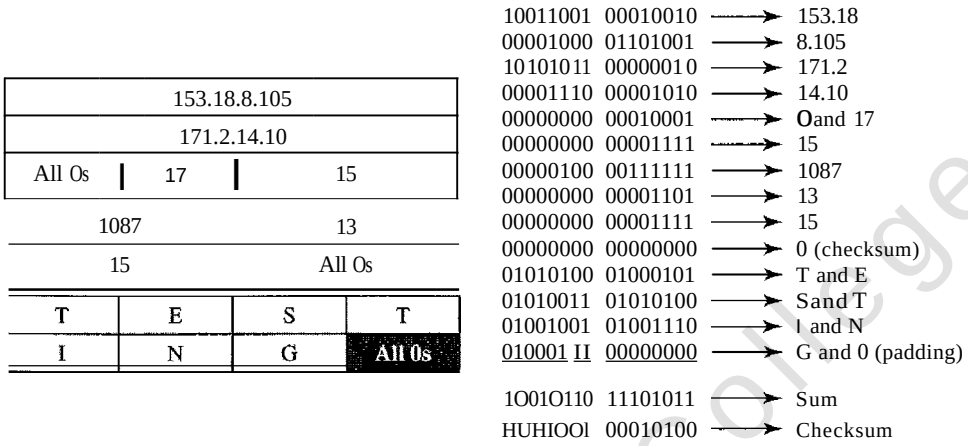
Example 23.2

Figure 23.11 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calcUlation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.

Optional Use of the Checksum

The calculation of the checksum and its inclusion in a user datagram are optional. If the checksum is not calculated, the field is filled with 1s. Note that a calculated checksum can never be allIs because this implies that the sum is all Os, which is impossible because it requires that the value of fields to be Os.

Figure 23.11 *Checksum calculation of a simple UDP user datagram*



UDP Operation

UDP uses concepts common to the transport layer. These concepts will be discussed here briefly, and then expanded in the next section on the TCP protocol.

Connectionless Services

As mentioned previously, UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered. Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.

One of the ramifications of being connectionless is that the process that uses UDP cannot send a stream of data to UDP and expect UDP to chop them into different related user datagrams. Instead each request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

Flow and Error Control

UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.

There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded.

The lack of flow control and error control means that the process using UDP should provide these mechanisms.

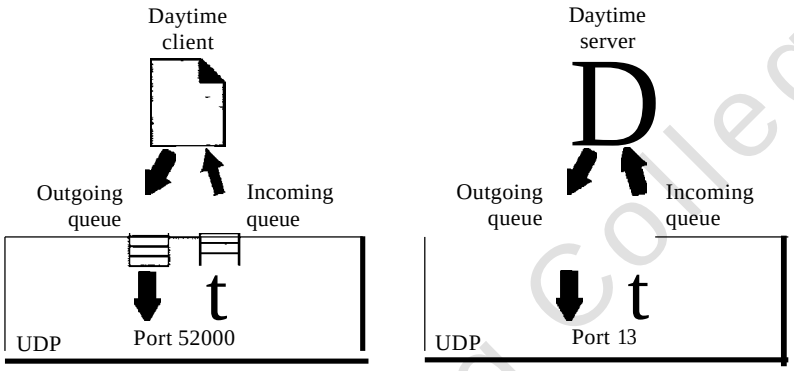
Encapsulation and Decapsulation

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Queuing

We have talked about ports without discussing the actual implementation of them. In UDP, queues are associated with ports (see Figure 23.12).

Figure 23.12 *Queues in UDP*



At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue associated with each process. Other implementations create only an incoming queue associated with each process.

Note that even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue. The queues opened by the client are, in most cases, identified by ephemeral port numbers. The queues function as long as the process is running. When the process terminates, the queues are destroyed.

The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow. If this happens, the operating system can ask the client process to wait before sending any more messages.

When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a *port unreachable* message to the server. All the incoming messages for one particular client program, whether coming from the same or a different server, are sent to the same queue. An incoming queue can overflow. If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent to the server.

At the server site, the mechanism of creating queues is different. In its simplest form, a server asks for incoming and outgoing queues, using its well-known port, when it starts running. The queues remain open as long as the server is running.

When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user

datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client. All the incoming messages for one particular server, whether coming from the same or a different client, are sent to the same queue. An incoming queue can overflow. If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent to the client.

When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow. If this happens, the operating system asks the server to wait before sending any more messages.

Use of UDP

The following lists some uses of the UDP protocol:

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FrP that needs to send bulk data (see Chapter 26).
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP (see Chapter 28).
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP) (see Chapter 22).

23.3 TCP

The second transport layer protocol we discuss in this chapter is called Transmission Control Protocol (TCP). TCP, like UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

In brief, TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.

TCP Services

Before we discuss TCP in detail, let us explain the services offered by TCP to the processes at the application layer.

Process-to-Process Communication

Like UDP, TCP provides process-to-process communication using port numbers. Table 23.2 lists some well-known port numbers used by TCP.

Table 23.2 *Well-known ports used by TCP*

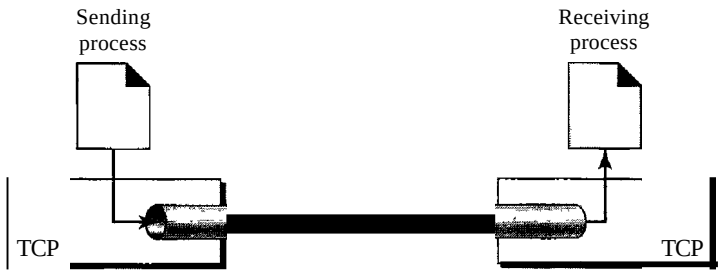
Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Stream Delivery Service

TCP, unlike UDP, is a stream-oriented protocol. In UDP, a process (an application program) sends messages, with predefined boundaries, to UDP for delivery. UDP adds its own header to each of these messages and delivers them to IP for transmission. Each message from the process is calIed a user datagram and becomes, eventually, one IP datagram. Neither IP nor UDP recognizes any relationship between the datagrams.

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet. This imaginary environment is depicted in Figure 23.13. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

Figure 23.13 *Stream delivery*



Sending and Receiving Buffers Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. (We will see later that these buffers are also necessary for flow and error control mechanisms used by TCP.) One way to implement a buffer is to use a circular array of I-byte locations as shown in Figure 23.14. For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.

Figure 23.14 *Sending and receiving buffers*

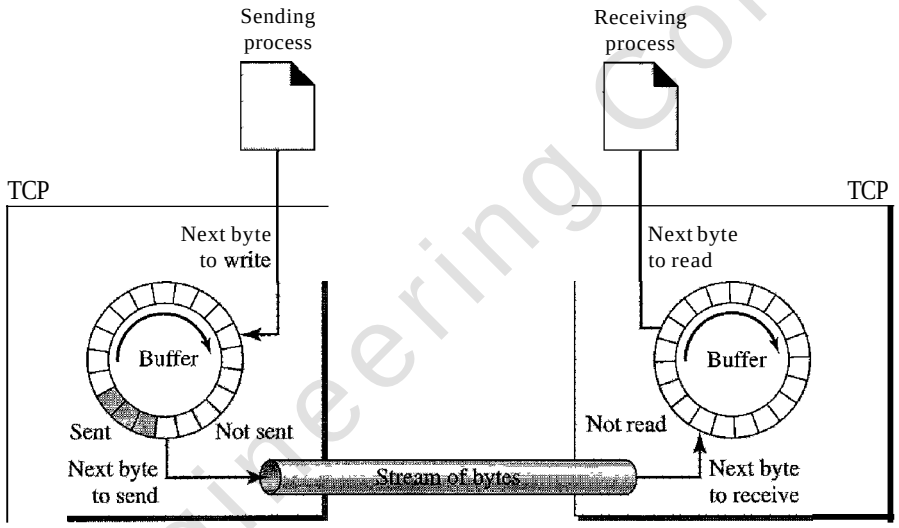


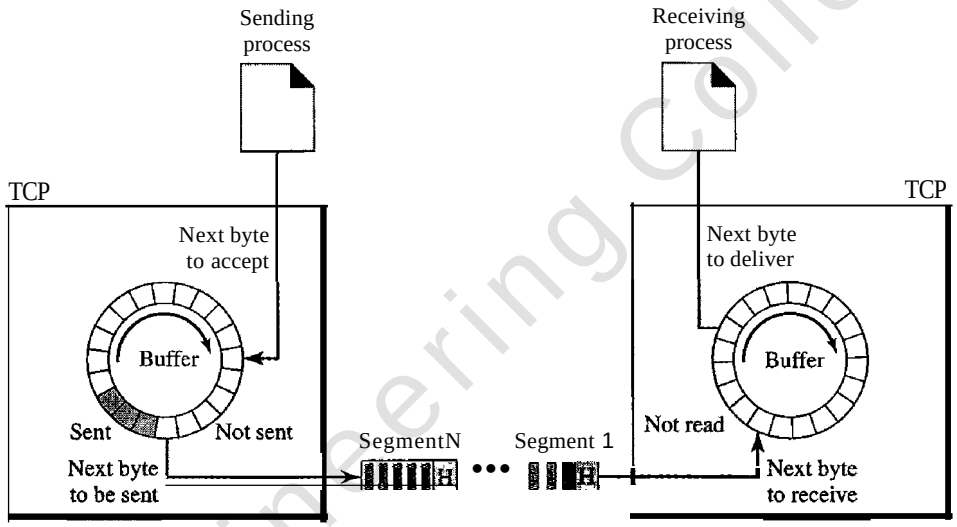
Figure 23.14 shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP. However, as we will see later in this chapter, TCP may be able to send only part of this colored section. This could be due to the slowness of the receiving process or perhaps to congestion in the network. Also note that after the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process. This is why we show a circular buffer.

The operation of the buffer at the receiver site is simpler. The circular buffer is divided into two areas (shown as white and colored). The white area contains empty chambers to be filled by bytes received from the network. The colored sections contain received bytes that can be read by the receiving process. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

Segments Although buffering handles the disparity between the speed of the producing and consuming processes, we need one more step before we can send data. The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At

the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagrams and transmitted. This entire operation is transparent to the receiving process. Later we will see that segments may be received out of order, lost, or corrupted and resent. All these are handled by TCP with the receiving process unaware of any activities. Figure 23.15 shows how segments are created from the bytes in the buffers.

Figure 23.15 *TCP segments*



Note that the segments are not necessarily the same size. In Figure 23.15, for simplicity, we show one segment carrying 3 bytes and the other carrying 5 bytes. In reality, segments carry hundreds, if not thousands, of bytes.

Full-Duplex Communication

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions.

Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Note that this is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent. Each may use a different path to reach the destination. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of

delivering the bytes in order to the other site. The situation is similar to creating a bridge that spans multiple islands and passing all the bytes from one island to another in one single connection. We will discuss this feature later in the chapter.

Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

TCP Features

To provide the services mentioned in the previous section, TCP has several features that are briefly summarized in this section and discussed later in detail.

Numbering System

Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.

Byte Number TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP generates a random number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.

Sequence Number After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

Example 23.3

Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 1O,00l. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

Solution

The following shows the sequence number for each segment:

Segment 1	Sequence Number: 10,001 (range: 10,001 to 11,(00)
Segment 2	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	Sequence Number: 14,001 (range: 14,001 to 15,000)

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

When a segment carries a combination of data and control information (piggybacking), it uses a sequence number. If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid. However, some segments, when carrying only control information, need a sequence number to allow an acknowledgment from the receiver. These segments are used for connection establishment, termination, or abortion. Each of these segments consumes one sequence number as though it carried 1 byte, but there are no actual data. If the randomly generated sequence number is x, the first data byte is numbered $x + 1$. The byte x is considered a phony byte that is used for a control segment to open a connection, as we will see shortly.

Acknowledgment Number As we discussed previously, communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time. Each party numbers the bytes, usually with a different starting byte number. The sequence number in each direction shows the number of the first byte carried by the segment. Each party also uses an acknowledgment number to confirm the bytes it has received. However, the acknowledgment number defines the number of the next byte that the party expects to receive. In addition, the acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds I to it, and announces this sum as the acknowledgment number. The term *cumulative* here means that if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642. Note that this does not mean that the party has received 5642 bytes because the first byte number does not have to start from 0.

The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

Flow Control

TCP, unlike UDP, provides *flow control.* The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

Error Control

To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented, as we will see later.

Congestion Control

TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also detennined by the level of congestion in the network.

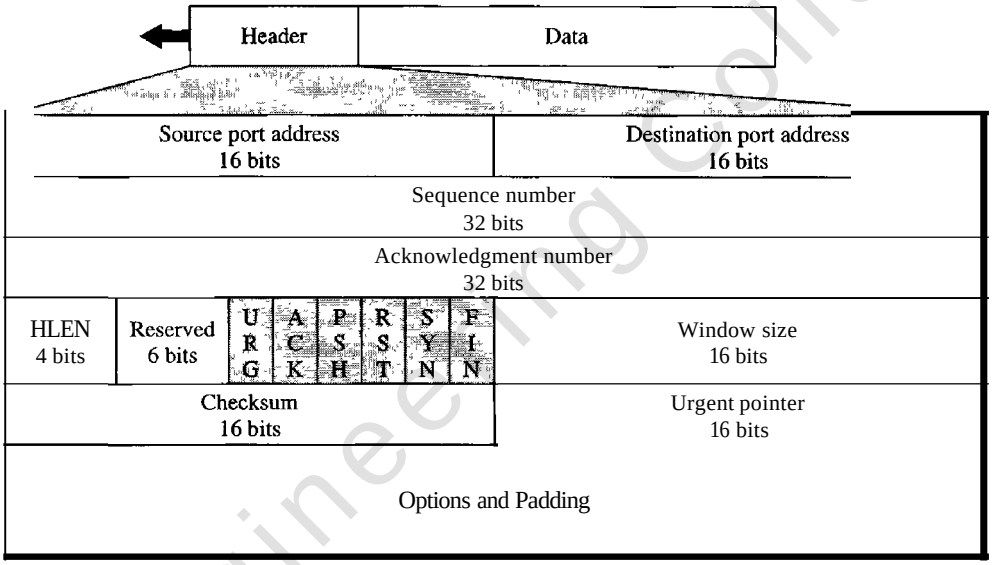
Segment

Before we discuss TCP in greater detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment.

Format

The format of a segment is shown in Figure 23.16.

Figure 23.16 *TCP segment format*



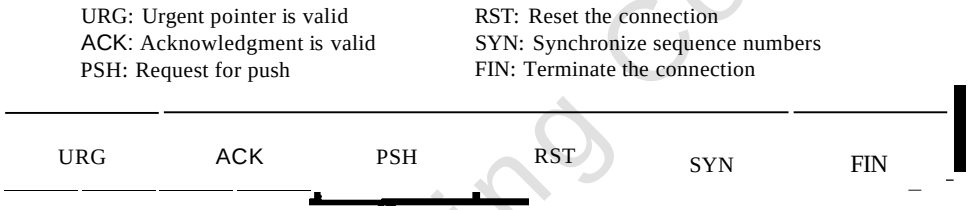
The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. We will discuss some of the header fields in this section. The meaning and purpose of these will become clearer as we proceed through the chapter.

- Source port address. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.
- Destination port address. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.
- Sequence number. This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.
- Acknowledgment number. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver

of the segment has successfully received byte number *x* from the other party, it defines *x* + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.

- Header length. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- Reserved. This is a 6-bit field reserved for future use.
- Control. This field defines 6 different control bits or flags as shown in Figure 23.17. One or more of these bits can be set at a time.

Figure 23.17 *Control field*



These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in Table 23.3. We will discuss them further when we study the detailed operation of TCP later in the chapter.

Table 23.3 *Description offlags in the control field*

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- Window size. This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- Checksum. This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same

purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.

- Urgent pointer. This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.
- Options. There can be up to 40 bytes of optional information in the TCP header. We will not discuss these options here; please refer to the reference list for more information.

A TCP Connection

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames. You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a TCP connection is virtual, not physical. TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted. Unlike TCP, IP is unaware of this retransmission. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.

In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

Connection Establishment

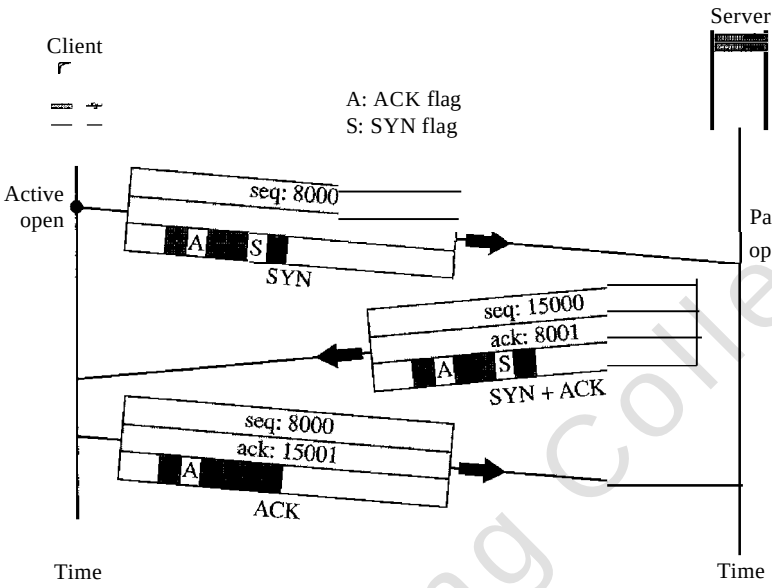
TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

Three-Way Handshaking The connection establishment in TCP is called three-way handshaking. In our example, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol.

The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a *passive open.* Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure 23.18. To show the process, we use two time lines: one at each site. Each segment has values for all its header fields and perhaps for some of its option fields, too. However, we show only the few fields necessary to understand each phase. We show the sequence number,

Figure 23.18 *Connection establishment using three-way handshaking*



the acknowledgment number, the control flags (only those that are set), and the window size, if not empty. The three steps in this phase are as follows.

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.

A SYN segment cannot carry data, but it consumes one sequence number.

---

2. The server sends the second segment, a SYN + ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

A SYN + ACK segment cannot carry data,
but does consume one sequence number.

---

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

An ACK segment, if carrying no data, consumes no sequence number.

Simultaneous Open A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

SYN Flooding Attack The connection establishment procedure in TCP is susceptible to a serious security problem called the SYN flooding attack. This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is corning from a different client by faking the source IP addresses in the datagrams. The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. The TCP server then sends the SYN + ACK segments to the fake clients, which are lost. During this time, however, a lot of resources are occupied without being used. If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash. This SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

Some implementations of TCP have strategies to alleviate the effects of a SYN attack. Some have imposed a limit on connection requests during a specified period of time. Others filter out datagrams coming from unwanted source addresses. One recent strategy is to postpone resource allocation until the entire connection is set up, using what is called a cookie. SCTP, the new transport layer protocol that we discuss in the next section, uses this strategy.

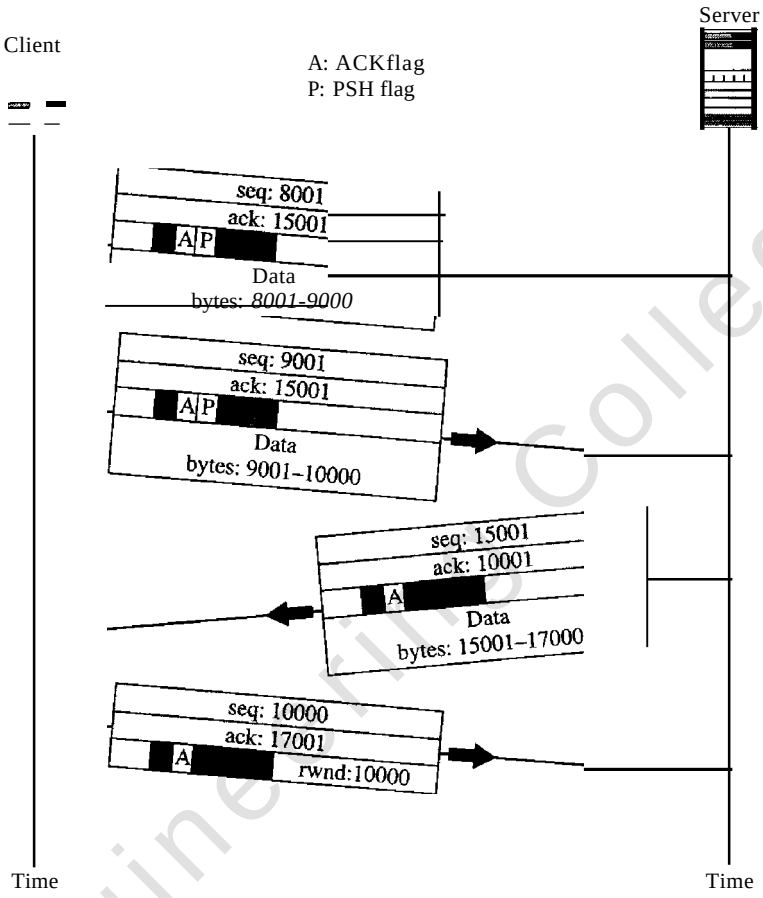
Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. We will study the rules of acknowledgment later in the chapter; for the moment, it is enough to know that data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data. Figure 23.19 shows an example. In this example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent. Note the values of the sequence and acknowledgment numbers. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received. We discuss the use of this flag in greater detail later. The segment from the server, on the other hand, does not set the push flag. Most TCP implementations have the option to set or not set this flag.

Pushing Data We saw that the sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size. The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP.

However, on occasion the application program has no need for this flexibility_ For example, consider an application program that communicates interactively with another

Figure 23.19 *Data transfer*



application program on the other end. The application program on one site wants to send a keystroke to the application at the other site and receive an immediate response. Delayed transmission and delayed delivery of data may not be acceptable by the application program.

TCP can handle such a situation. The application program at the sending site can request a *push* operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

Although the push operation can be requested by the application program, most current implementations ignore such requests. TCP can choose whether or not to use this feature.

Urgent Data TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, on occasion an application program needs to send *urgent* bytes. This means that the sending application program wants a piece of data to be read out of order by the receiving application program. As an example, suppose that the sending

application program is sending data to be processed by the receiving application program. When the result of processing comes back, the sending application program finds that everything is wrong. It wants to abort the process, but it has already sent a huge amount of data. If it issues an abort command (control + C), these two characters will be stored at the end of the receiving TCP buffer. It will be delivered to the receiving application program after all the data have been processed.

The solution is to send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer. The urgent pointer field in the header defines the end of the urgent data and the start of normal data.

When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

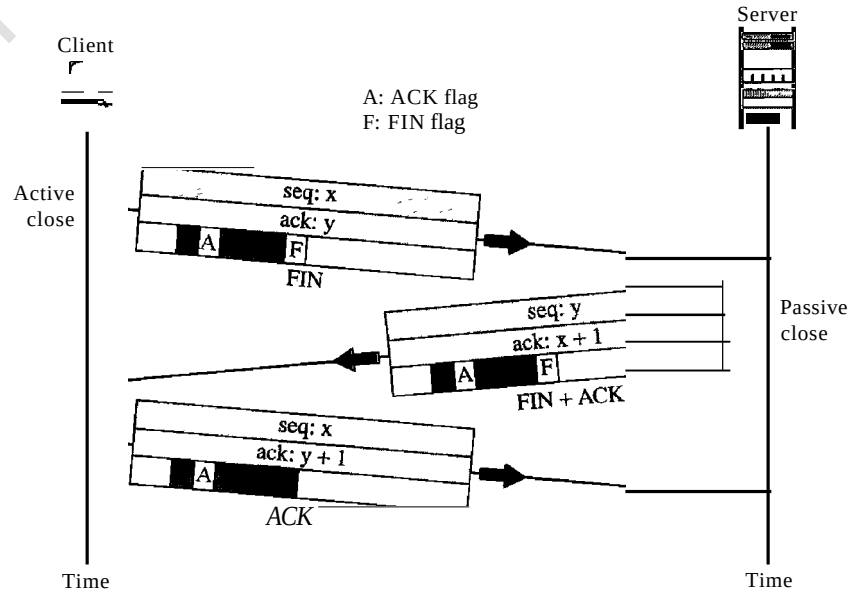
Connection Termination

Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client. Most implementations today allow two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three-Way Handshaking Most implementations today allow *three-way handshaking* for connection termination as shown in Figure 23.20.

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it

Figure 23.20 *Connection termination using three-way handshaking*



can be just a control segment as shown in Figure 23.20. If it is only a control segment, it consumes only one sequence number.

The FIN segment consumes one sequence number if it does not carry data.

---

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

The FIN + ACK segment consumes one sequence number if it does not carry data.

---

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Half-Close In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin. A good example is sorting. When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all the data, can close the connection in the outbound direction. However, the inbound direction must remain open to receive the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open.

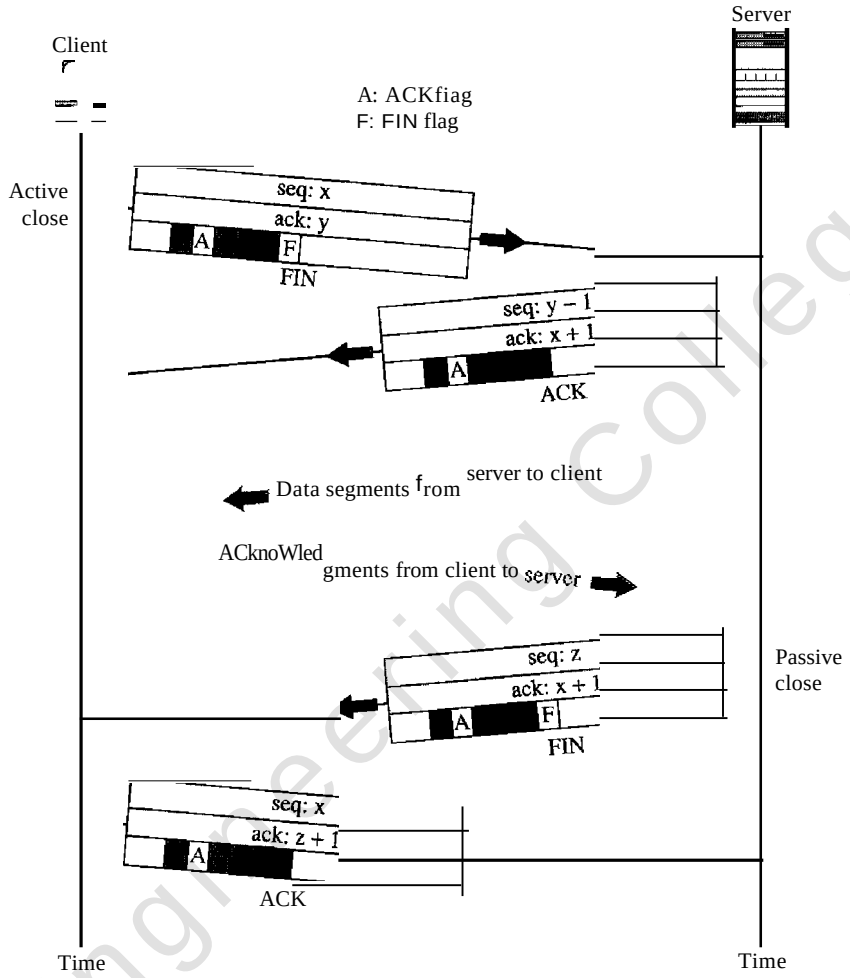
Figure 23.21 shows an example of a half-close. The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops. The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server. Note the sequence numbers we have used. The second segment (ACK) consumes no sequence number. Although the client has received sequence number $y - 1$ and is expecting y, the server sequence number is still $y - 1$. When the connection finally closes, the sequence number of the last ACK segment is still x, because no sequence numbers are consumed during data transfer in that direction.

Flow Control

TCP uses a sliding window, as discussed in Chapter 11, to handle flow control. The sliding window protocol used by TCP, however, is something between the *Go-Back-N* and Selective Repeat sliding window. The sliding window protocol in TCP looks like

Figure 23.21 *Half-close*



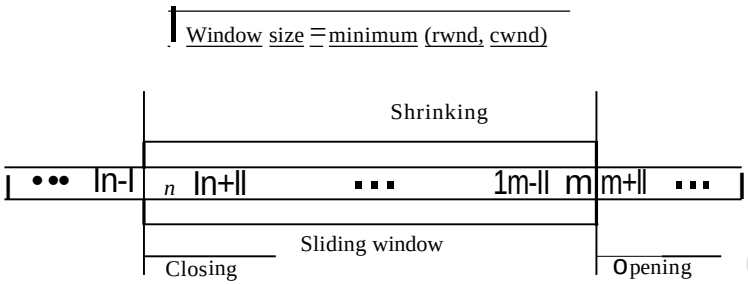
the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive. There are two big differences between this sliding window and the one we used at the data link layer. First, the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented. Second, the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size.

Figure 23.22 shows the sliding window in TCP. The window spans a portion of the buffer containing bytes received from the process. The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment. The imaginary window has two walls: one left and one right.

The window is *opened, closed,* or *shrunk.* These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender

Figure 23.22 *Sliding window*



need not worry about them anymore. Shrinking the window means moving the right wall to the left. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending. This is a problem if the sender has already sent these bytes. Note that the left wall cannot move to the left because this would revoke some of the previously sent acknowledgments.

A sliding window is used to make transmission more efficient as weD as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented.

The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd)* or *congestion window (cwnd).* The *receiver window* is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded. The congestion window is a value determined by the network to avoid congestion. We will discuss congestion later in the chapter.

Example 23.4

What is the value of the receiver window *(rwnd)* for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?

Solution

The value of *rwnd* = 5000 - 1000 = 4000. Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.

Example 23.5

What is the size of the window for host A if the value of *rwnd* is 3000 bytes and the value of *cwnd* is 3500 bytes?

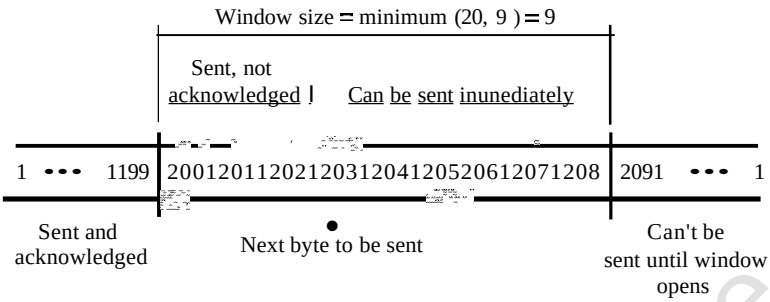
Solution

The size of the window is the smaller of *rwnd* and *cwnd,* which is 3000 bytes.

Example 23.6

Figure 23.23 shows an unrealistic example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20 (in reality this value is thousands of bytes). The receiver has sent

Figure 23.23 *Example 23.6*



an acknowledgment number of 200 with an *rwnd* of 9 bytes (in reality this value is thousands of bytes). The size of the sender window is the minimum of *rwnd* and *cwnd,* or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.

Some points about TCP sliding windows:

- The size of the window is the lesser of *rwnd* and *cwnd.*
- The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Error Control

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

Checksum

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment. We will see, in Chapter 24, that the 16-bit checksum is considered inadequate for the new transport

layer, SCTP. However, it cannot be changed for TCP because this would involve reconfiguration of the entire header format.

Acknowledgment

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

ACK segments do not consume sequence numbers and are not acknowledged.

Retransmission

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted. In modern implementations, a segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs.

In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.

Note that no retransmission occurs for segments that do not consume sequence numbers. In particular, there is no transmission for an ACK segment.

No retransmission timer is set for an ACK segment.

Retransmission After RTO A recent implementation of TCP maintains one retransmission time-out (RTO) timer for all outstanding (sent, but not acknowledged) segments. When the timer matures, the earliest outstanding segment is retransmitted even though lack of a received ACK can be due to a delayed segment, a delayed ACK, or a lost acknowledgment. Note that no time-out timer is set for a segment that carries only an acknowledgment, which means that no such segment is resent. The value of RTO is dynamic in TCP and is updated based on the round-trip time (RTT) of segments. An RTI is the time needed for a segment to reach a destination and for an acknowledgment to be received. It uses a back-off strategy similar to one discussed in Chapter 12.

Retransmission After Three Duplicate ACK Segments The previous rule about retransmission of a segment is sufficient if the value of RTO is not very large. Sometimes, however, one segment is lost and the receiver receives so many out-of-order segments that they cannot be saved (limited buffer size). To alleviate this situation, most implementations today follow the three-duplicate-ACKs rule and retransmit the missing segment immediately. This feature is referred to as fast retransmission, which we will see in an example shortly.

Out-oj-Order Segments

When a segment is delayed, lost, or discarded, the segments following that segment arrive out of order. Originally, TCP was designed to discard all out-of-order segments, resulting

in the retransmission of the missing segment and the following segments. Most implementations today do not discard the out-of-order segments. They store them temporarily and flag them as out-of-order segments until the missing segment arrives. Note, however, that the out-of-order segments are not delivered to the process. TCP guarantees that data are delivered to the process in order.

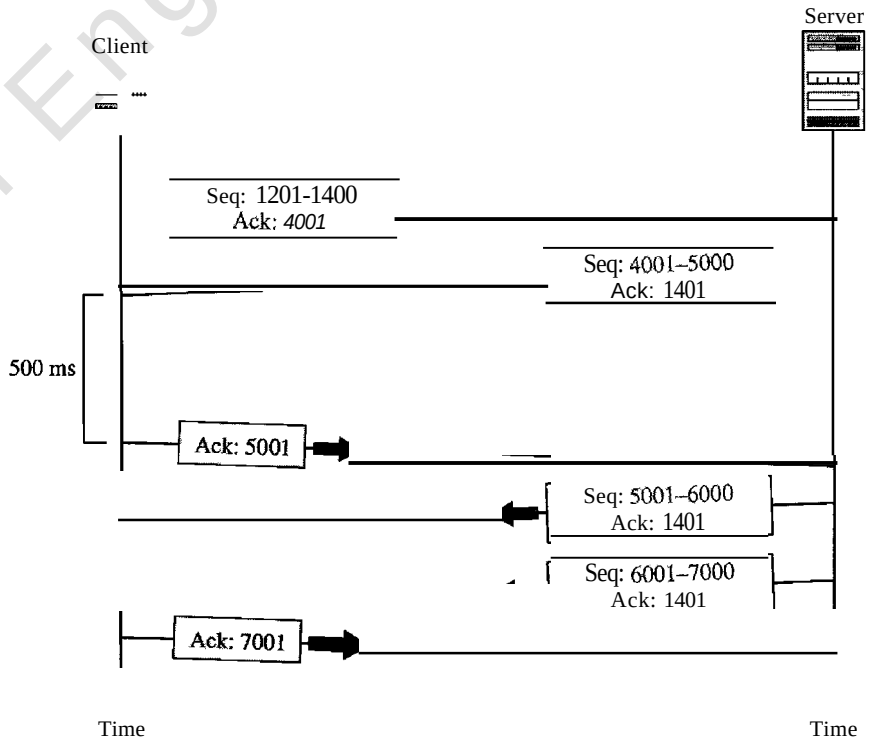
Data may arrive out of order and be temporarily stored by the receiving TCP, but yep guarantees that no out-of-order segment is delivered to the process.

Some Scenarios

In this section we give some examples of scenarios that occur during the operation of TCP. In these scenarios, we show a segment by a rectangle. If the segment carries data, we show the range of byte numbers and the value of the acknowledgment field. If it carries only an acknowledgment, we show only the acknowledgment number in a smaller box.

Normal Operation The first scenario shows bidirectional data transfer between two systems, as in Figure 23.24. The client TCP sends one segment; the server TCP sends three. The figure shows which rule applies to each acknowledgment. There are data to be sent, so the segment displays the next byte expected. When the client receives the first segment from the server, it does not have any more data to send; it sends only an

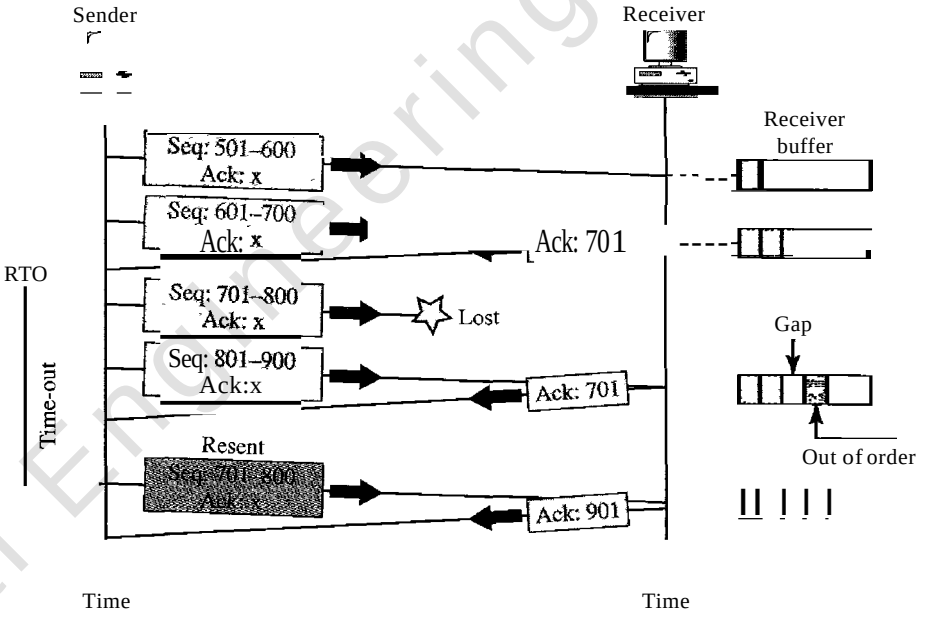
Figure 23.24 *Nonnaloperation*



ACK segment. However, the acknowledgment needs to be delayed for 500 ms to see if any more segments arrive. When the timer matures, it triggers an acknowledgment. This is so because the client has no knowledge if other segments are coming; it cannot delay the acknowledgment forever. When the next segment arrives, another acknowledgment timer is set. However, before it matures, the third segment arrives. The arrival of the third segment triggers another acknowledgment.

Lost Segment In this scenario, we show what happens when a segment is lost or corrupted. A lost segment and a corrupted segment are treated the same way by the receiver. A lost segment is discarded somewhere in the network; a corrupted segment is discarded by the receiver itself. Both are considered lost. Figure 23.25 shows a situation in which a segment is lost and discarded by some router in the network, perhaps due to congestion.

Figure 23.25 *Lost segment*



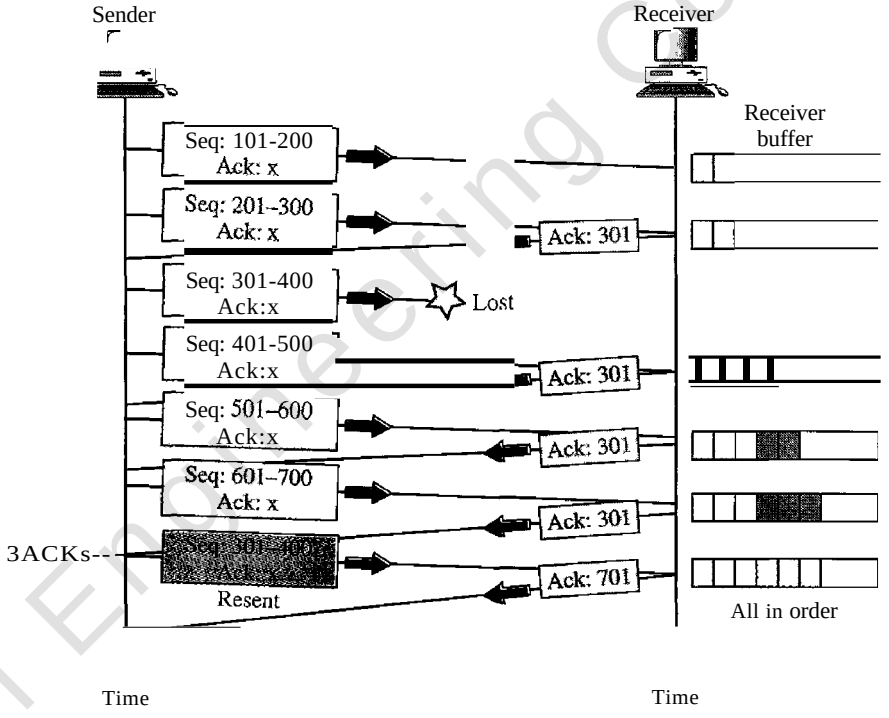
We are assuming that data transfer is unidirectional: one site is sending, the other is receiving. In our scenario, the sender sends segments 1 and 2, which are acknowledged immediately by an ACK. Segment 3, however, is lost. The receiver receives segment 4, which is out of order. The receiver stores the data in the segment in its buffer but leaves a gap to indicate that there is no continuity in the data. The receiver immediately sends an acknowledgment to the sender, displaying the next byte it expects. Note that the receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.

The receiver TCP delivers only ordered data to the process.

We have shown the timer for the earliest outstanding segment. The timer for this definitely runs out because the receiver never sends an acknowledgment for lost or out-of-order segments. When the timer matures, the sending TCP resends segment 3, which arrives this time and is acknowledged properly. Note that the value in the second and third acknowledgments differs according to the corresponding rule.

Fast Retransmission **In** this scenario, we want to show the idea of fast retransmission. Our scenario is the same as the second except that the RTO has a higher value (see Figure 23.26).

Figure 23.26 *Fast retransmission*



When the receiver receives the fourth, fifth, and sixth segments, it triggers an acknowledgment. The sender receives four acknowledgments with the same value (three duplicates). Although the timer for segment 3 has not matured yet, the fast transmission requires that segment 3, the segment that is expected by all these acknowledgments, be resent immediately.

Note that only one segment is retransmitted although four segments are not acknowledged. When the sender receives the retransmitted ACK, it knows that the four segments are safe and sound because acknowledgment is cumulative.

Congestion Control

We discuss congestion control of TCP in Chapter 24.

23.4 SCTP

Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced. These new applications, such as IUA (ISDN over IP), M2UA and M3UA (telephony signaling), H.248 (media gateway control), H.323 (IP telephony), and SIP (IP telephony), need a more sophisticated service than TCP can provide. SCTP provides this enhanced performance and reliability. We briefly compare UDP, TCP, and SCTP:

- UDP is a message-oriented protocol. A process delivers a message to UDP, which is encapsulated in a user datagram and sent over the network. UDP *conserves the message boundaries;* each message is independent of any other message. This is a desirable feature when we are dealing with applications such as IP telephony and transmission of real-time data, as we will see later in the text. However, UDP is unreliable; the sender cannot know the destiny of messages sent. A message can be lost, duplicated, or received out of order. UDP also lacks some other features, such as congestion control and flow control, needed for a friendly transport layer protocol.
- TCP is a byte-oriented protocol. It receives a message or messages from a process, stores them as a stream of bytes, and sends them in segments. There is no preservation of the message boundaries. However, TCP is a reliable protocol. The duplicate segments are detected, the lost segments are resent, and the bytes are delivered to the end process in order. TCP also has congestion control and flow control mechanisms.
- SCTP combines the best features of UDP and TCP. SCTP is a reliable message-oriented protocol. It preserves the message boundaries and at the same time detects lost data, duplicate data, and out-of-order data. It also has congestion control and flow control mechanisms. Later we will see that SCTP has other innovative features unavailable in UDP and TCP.

SCTP is a *message-oriented, reliable* protocol that combines the best features of UDP and TCP.

SCTP Services

Before we discuss the operation of SCTP, let us explain the services offered by SCTP to the application layer processes.

Process-to-Process Communication

SCTP uses all well-known ports in the TCP space. Table 23.4 lists some extra port numbers used by SCTP.

Multiple Streams

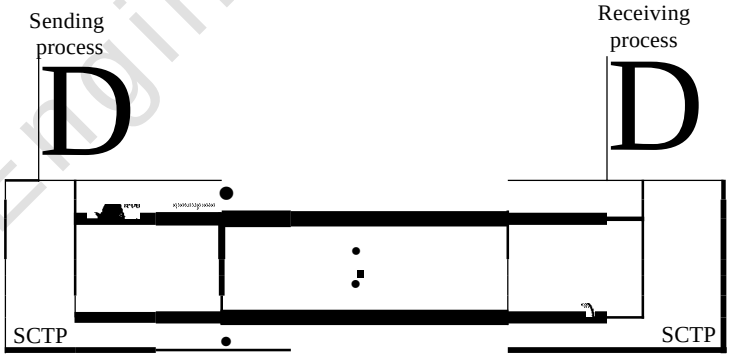
We learned in the previous section that TCP is a stream-oriented protocol. Each connection between a TCP client and a TCP server involves one single stream. The problem

Table 23.4 *Some SCTP applications*

Protocol	*Port Number*	*Description*
IVA	9990	ISDN overIP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718,1719, 1720, 11720	IP telephony
SIP	5060	IP telephony

with this approach is that a loss at any point in the stream blocks the delivery of the rest of the data. This can be acceptable when we are transferring text; it is not when we are sending real-time data such as audio or video. SCTP allows multistream service in each connection, which is called association in SCTP terminology. If one of the streams is blocked, the other streams can still deliver their data. The idea is similar to multiple lanes on a highway. Each lane can be used for a different type of traffic. For example, one lane can be used for regular traffic, another for car pools. If the traffic is blocked for regular vehicles, car pool vehicles can still reach their destinations. Figure 23.27 shows the idea of multiple-stream delivery.

Figure 23.27 *Multiple-stream concept*



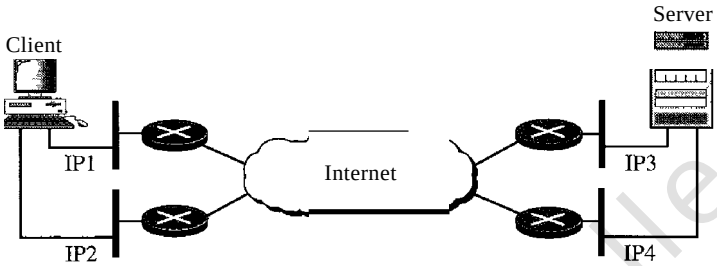
An association in SCTP can involve multiple streams.

Multihoming

A TCP connection involves one source and one destination IP address. This means that even if the sender or receiver is a multihomed host (connected to more than one physical address with multiple IP addresses), only one of these IP addresses per end can be utilized during the connection. An SCTP association, on the other hand, supports multihoming service. The sending and receiving host can define multiple IP addresses in each end for an association. In this fault-tolerant approach, when one path fails, another interface can be used for data delivery without interruption. This fault-tolerant

feature is very helpful when we are sending and receiving a real-time payload such as Internet telephony. Figure 23.28 shows the idea of multihoming.

Figure 23.28 *Multihoming concept*



In Figure 23.28, the client is connected to two local networks with two IP addresses. The server is also connected to two networks with two IP addresses. The client and the server can make an association, using four different pairs of IP addresses. However, note that in the current implementations of SCTP, only one pair of IF addresses can be chosen for normal communication; the alternative is used if the main choice fails. In other words, at present, SCTP does not allow load sharing between different paths.

SCTP association allows multiple IP addresses for each end.

Full-Duplex Communication

Like TCP, SCTP offers full-duplex service, in which data can flow in both directions at the same time. Each SCTP then has a sending and receiving buffer, and packets are sent in both directions.

Connection-Oriented Service

Like TCP, SCTP is a connection-oriented protocol. However, in SCTP, a connection is called an association. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two SCTPs establish an association between each other.
2. Data are exchanged in both directions.
3. The association is terminated.

Reliable Service

SCTP, like TCP, is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

SCTP Features

Let us first discuss the general features of SCTP and then compare them with those of TCP.

Transmission Sequence Number

The unit of data in TCP is a byte. Data transfer in TCP is controlled by numbering bytes by using a sequence number. On the other hand, the unit of data in SCTP is a DATA chunk which mayor may not have a one-to-one relationship with the message coming from the process because of fragmentation (discussed later), Data transfer in SCTP is controlled by numbering the data chunks. SCTP uses a transmission sequence number (TSN) to number the data chunks. In other words, the TSN in SCTP plays the analogous role to the sequence number in TCP. TSNs are 32 bits long and randomly initialized between 0 and $2^{32} - 1$. Each data chunk must carry the corresponding TSN in its header.

In **SCTP,** a data chunk is numbered using a TSN.

Stream Identifier

In TCP, there is only one stream in each connection. In SCTP, there may be several streams in each association. Each stream in SCTP needs to be identified by using a stream identifier (SI). Each data chunk must carry the SI in its header so that when it arrives at the destination, it can be properly placed in its stream. The 51 is a 16-bit number starting from 0.

To distinguish between different streams, SCTP uses an SI.

Stream Sequence Number

When a data chunk arrives at the destination SCTP, it is delivered to the appropriate stream and in the proper order. This means that, in addition to an SI, SCTP defines each data chunk in each stream with a stream sequence number (SSN).

To distinguish between different data chunks belonging to the same stream, SCTP uses SSNs.

Packets

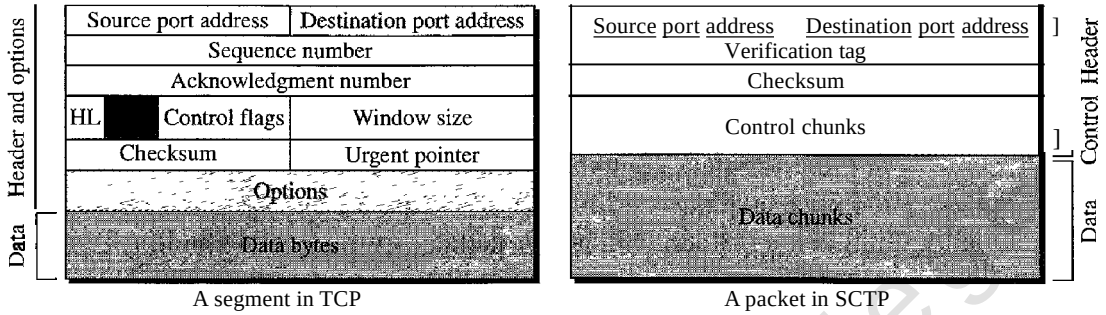
In TCP, a segment carries data and control information. Data are carried as a collection of bytes; control information is defined by six control flags in the header. The design of SCTP is totally different: data are carried as data chunks, control information is carried as control chunks. Several control chunks and data chunks can be packed together in a packet. A packet in SCTP plays the same role as a segment in TCP. Figure 23.29 compares a segment in TCP and a packet in SCTP. Let us briefly list the differences between an SCTP packet and a TCP segment:

TCP has segments; SCTP has packets.

---

1. The control information in TCP is part of the header; the control information in SCTP is included in the control chunks. There are several types of control chunks; each is used for a different purpose.

Figure 23.29 *Comparison between a TCP segment and an SCTP packet*



2. The data in a TCP segment treated as one entity; an SCTP packet can carry several data chunks; each can belong to a different stream.
3. The options section, which can be part of a TCP segment, does not exist in an SCTP packet. Options in SCTP are handled by defining new chunk types.
4. The mandatory part of the TCP header is 20 bytes, while the general header in SCTP is only 12 bytes. The SCTP header is shorter due to the following:
   a. An SCTP sequence number (TSN) belongs to each data chunk and hence is located in the chunk's header.
   b. The acknowledgment number and window size are part of each control chunk.
   c. There is no need for a header length field (shown as HL in the TCP segment) because there are no options to make the length of the header variable; the SCTP header length is fixed (12 bytes).
   d. There is no need for an urgent pointer in SCTP.
5. The checksum in TCP is 16 bits; in SCTP, it is 32 bits.
6. The verification tag in SCTP is an association identifier, which does not exist in TCP. In TCP, the combination of IP and port addresses defines a connection; in SCTP we may have multihorning using different IP addresses. A unique verification tag is needed to define each association.
7. TCP includes one sequence number in the header, which defines the number of the first byte in the data section. An SCTP packet can include several different data chunks. TSNs, SIs, and SSNs define each data chunk.
8. Some segments in TCP that carry control information (such as SYN and FIN) need to consume one sequence number; control chunks in SCTP never use a TSN, SI, or SSN. These three identifiers belong only to data chunks, not to the whole packet.

In SCTP, control information and data information are carried in separate chunks.

In SCTP, we have data chunks, streams, and packets. An association may send many packets, a packet may contain several chunks, and chunks may belong to different streams. To make the definitions of these terms clear, let us suppose that process A needs to send 11 messages to process B in three streams. The first four messages are in

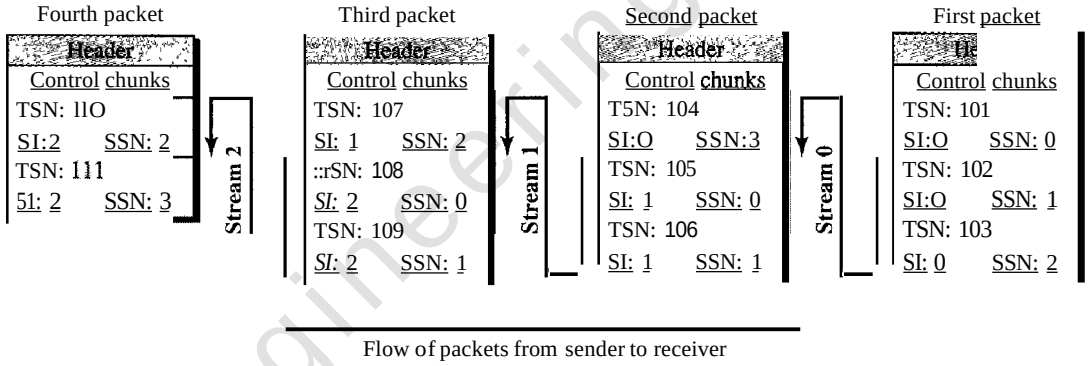
the first stream, the second three messages are in the second stream, and the last four messages are in the third stream.

Although a message, if long, can be carried by several data chunks, we assume that each message fits into one data chunk. Therefore, we have 11 data chunks in three streams.

The application process delivers 11 messages to SCTP, where each message is ear-marked for the appropriate stream. Although the process could deliver one message from the first stream and then another from the second, we assume that it delivers all messages belonging to the first stream first, all messages belonging to the second stream next, and finally, all messages belonging to the last stream.

We also assume that the network allows only three data chunks per packet, which means that we need. four packets as shown in Figure 23.30. Data chunks in stream 0 are carried in the first packet and part of the second packet; those in stream 1 are carried in the second and third packets; those in stream 2 are carried in the third and fourth packets.

Figure 23.30 *Packet, data chunks, and streams*



Note that each data chunk needs three identifiers: TSN, SI, and SSN. TSN is a cumulative number and is used, as we will see later, for flow control and error control. SI defines the stream to which the chunk belongs. SSN defines the chunk's order in a particular stream. In our example, SSN starts from 0 for each stream.

Data chunks are identified by three items: TSN, SI, and SSN.
TSN is a cumulative number identifying the association;
SI defines the stream; SSN defines the chunk in a stream.

Acknowledgment Number

TCP acknowledgment numbers are byte-oriented and refer to the sequence numbers. SCTP acknowledgment numbers are chunk-oriented. They refer to the TSN. A second difference between TCP and SCTP acknowledgments is the control information. Recall that this information is part of the segment header in TCP. To acknowledge segments that carry only control information, TCP uses a sequence number and acknowledgment number (for example, a SYN segment needs to be acknowledged by an ACK segment). In SCTP, however, the control information is carried by control chunks, which do not

need a TSN. These control chunks are acknowledged by another control chunk of the appropriate type (some need no acknowledgment). For example, an INIT control chunk is acknowledged by an INIT ACK chunk. There is no need for a sequence number or an acknowledgment number.

In **SCTP,** acknowledgment numbers are used to acknowledge only data chunks; control chunks are acknowledged by other control chunks **if** necessary.

Flow Control

Like TCP, SCTP implements flow control to avoid overwhelming the receiver. We will discuss SCTP flow control later in the chapter.

Error Control

Like TCP, SCTP implements error control to provide reliability. TSN numbers and acknowledgment numbers are used for error control. We will discuss error control later in the chapter.

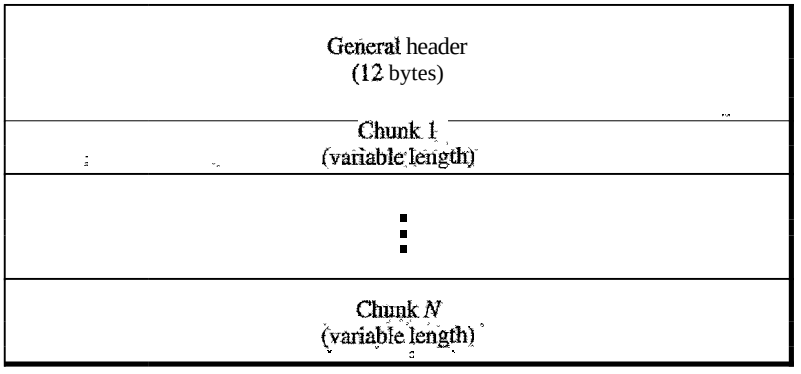
Congestion Control

Like TCP, SCTP implements congestion control to determine how many data chunks can be injected into the network. We will discuss congestion control in Chapter 24.

Packet Format

In this section, we show the format of a packet and different types of chunks. Most of the information presented in this section will become clear later; this section can be skipped in the first reading or used only as a reference. An SCTP packet has a mandatory general header and a set of blocks called chunks. There are two types of chunks: control chunks and data chunks. A control chunk controls and maintains the association; a data chunk carries user data. In a packet, the control chunks corne before the data chunks. Figure 23.31 shows the general format of an SCTP packet.

Figure 23.31 *SCTP packet format*

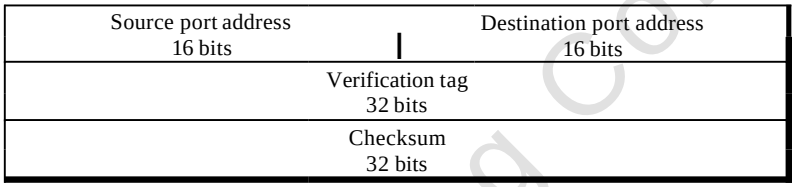


In an SCTP packet, control chunks come before data chunks.

General Header

The general header (packet header) defines the endpoints of each association to which the packet belongs, guarantees that the packet belongs to a particular association, and preserves the integrity of the contents of the packet including the header itself. The fonnat of the general header is shown in Figure 23.32.

Figure 23.32 *General header*



There are four fields in the general header:

- Source port address. This is a 16-bit field that defines the port number of the process sending the packet.
- Destination port address. This is a 16-bit field that defines the port number of the process receiving the packet.
- Verification tag. This is a number that matches a packet to an association. This prevents a packet from a previous association from being mistaken as a packet in this association. It serves as an identifier for the association; it is repeated in every packet during the association. There is a separate verification used for each direction in the association.
- Checksum. This 32-bit field contains a CRC-32 checksum. Note that the size of the checksum is increased from 16 (in UDP, TCP, and IP) to 32 bits to allow the use of the CRC-32 checksum.

Chunks

Control infonnation or user data are carried in chunks. The detailed fonnat of each chunk is beyond the scope of this book. See [For06] for details. The first three fields are common to all chunks; the information field depends on the type of chunk. The important point to remember is that SCTP requires the information section to be a multiple of 4 bytes; if not, padding bytes (eight as) are added at the end of the section. See Table 23.5 for a list of chunks and their descriptions.

An SCTP Association

SCTP, like TCP, is a connection-oriented protocol. However, a connection in SCTP is called an *association* to emphasize multihoming.

Table 23.5 *Chunks*

Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INITACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Aborts an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIEACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARDTSN	For adjusting cumulative TSN

A connection in SCTP is called an association.

Association Establishment

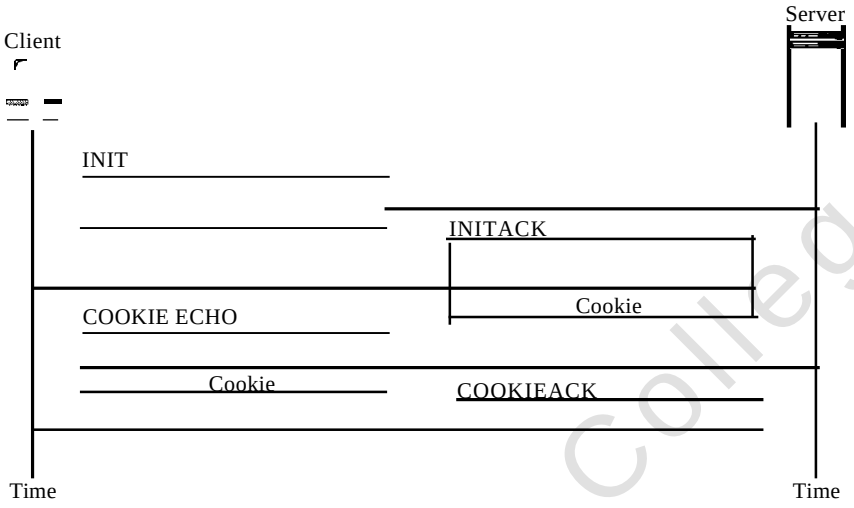
Association establishment in SCTP requires a four-way handshake. In this procedure, a process, normally a client, wants to establish an association with another process, normally a server, using SCTP as the transport layer protocol. Similar to TCP, the SCTP server needs to be prepared to receive any association (passive open). Association establishment, however, is initiated by the client (active open). SCTP association establishment is shown in Figure 23.33. The steps, in a normal situation, are as follows:

1. The client sends the first packet, which contains an INIT chunk.
2. The server sends the second packet, which contains an INIT ACK chunk.
3. The client sends the third packet, which includes a COOKIE ECHO chunk. This is a very simple chunk that echoes, without change, the cookie sent by the server. SCTP allows the inclusion of data chunks in this packet.
4. The server sends the fourth packet, which includes the COOKIE ACK chunk that acknowledges the receipt of the COOKIE ECHO chunk. SCTP allows the inclusion of data chunks with this packet.

No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk. A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.

Cookie We discussed a SYN flooding attack in the previous section. With TCP, a malicious attacker can flood a TCP server with a huge number of phony SYN segments using different forged IP addresses. Each time the server receives a SYN segment, it

Figure 23.33 *Four-way handshaking*



sets up a state table and allocates other resources while waiting for the next segment to arrive. After a while, however, the server may collapse due to the exhaustion of resources.

The designers of SCTP have a strategy to prevent this type of attack. The strategy is to postpone the allocation of resources until the reception of the third packet, when the IP address of the sender is verified. The information received in the first packet must somehow be saved until the third packet arrives. But if the server saved the information, that would require the allocation of resources (memory); this is the dilemma. The solution is to pack the information and send it back to the client. This is called generating a cookie. The cookie is sent with the second packet to the address received in the first packet. There are two potential situations.

1. If the sender of the first packet is an attacker, the server never receives the third packet; the cookie is lost and no resources are allocated. The only effort for the server is "baking" the cookie.
2. If the sender of the first packet is an honest client that needs to make a connection, it receives the second packet, with the cookie. It sends a packet (third in the series) with the cookie, with no changes. The server receives the third packet and knows that it has come from an honest client because the cookie that the sender has sent is there. The server can now allocate resources.

The above strategy works if no entity can "eat" a cookie "baked" by the server. To guarantee this, the server creates a digest (see Chapter 30) from the information, using its own secret key. The information and the digest together make the cookie, which is sent to the client in the second packet. When the cookie is returned in the third packet, the server calculates the digest from the information. If the digest matches the one that is sent, the cookie has not been changed by any other entity.

Data Transfer

The whole purpose of an association is to transfer data between two ends. After the association is established, bidirectional data transfer can take place. The client and the server can both send data. Like TCP, SCTP supports piggybacking.

There is a major difference, however, between data transfer in TCP and SCTP. TCP receives messages from a process as a stream of bytes without recognizing any boundary between them. The process may insert some boundaries for its peer use, but TCP treats that mark as part of the text. In other words, TCP takes each message and appends it to its buffer. A segment can carry parts of two different messages. The only ordering system imposed by TCP is the byte numbers.

SCTP, on the other hand, recognizes and maintains boundaries. Each message coming from the process is treated as one unit and inserted into a DATA chunk unless it is fragmented (discussed later). In this sense, SCTP is like UDP, with one big advantage: data chunks are related to each other.

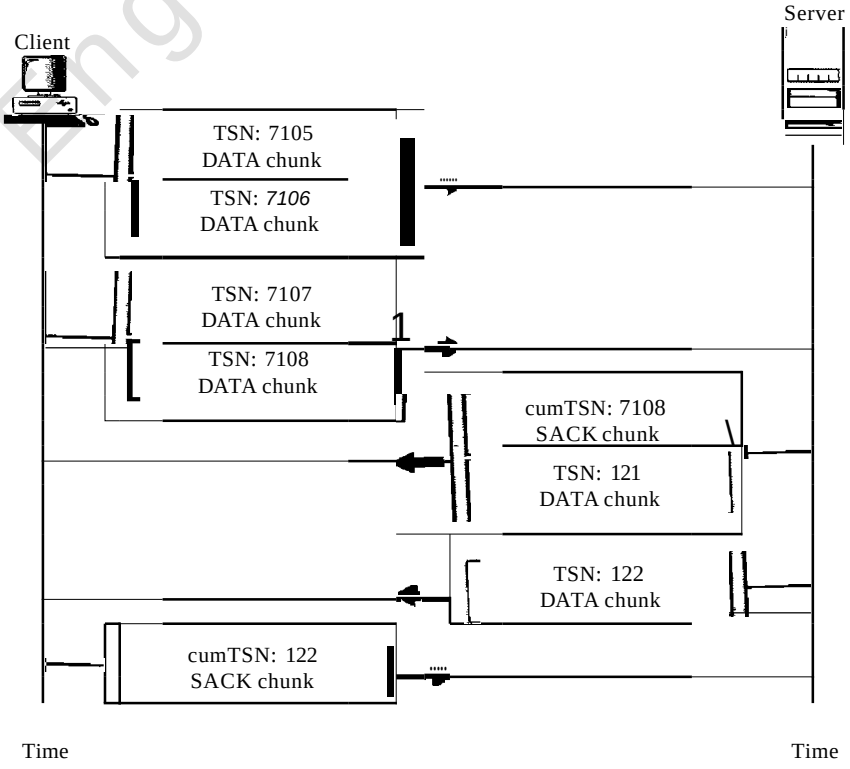
A message received from a process becomes a DATA chunk, or chunks if fragmented, by adding a DATA chunk header to the message. Each DATA chunk formed by a message or a fragment of a message has one TSN. We need to remember that only DATA chunks use TSNs and only DATA chunks are acknowledged by SACK chunks.

In SCTP, only DATA chunks consume TSNs;
DATA chunks are the only chunks that are acknowledged.

Let us show a simple scenario in Figure 23.34. In this figure a client sends four DATA chunks and receives two DATA chunks from the server. Later, we will discuss

---

Figure 23.34 *Simple data transfer*



the use of flow and error control in SCTP. For the moment, we assume that everything goes well in this scenario.

1. The client sends the first packet carrying two DATA chunks with TSNs 7105 and 7106.
2. The client sends the second packet carrying two DATA chunks with TSNs 7107 and 7108.
3. The third packet is from the server. It contains the SACK chunk needed to acknowledge the receipt of DATA chunks from the client. Contrary to TCP, SCTP acknowledges the last in-order TSN received, not the next expected. The third packet also includes the first DATA chunk from the server with TSN 121.
4. After a while, the server sends another packet carrying the last DATA chunk with TSN 122, but it does not include a SACK chunk in the packet because the last DATA chunk received from the client was already acknowledged.
5. Finally, the client sends a packet that contains a SACK chunk acknowledging the receipt of the last two DATA chunks from the server.

The acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.

Multihoming Data Transfer We discussed the multihoming capability of SCTP, a feature that distinguishes SCTP from UDP and TCP. Multihoming allows both ends to define multiple IP addresses for communication. However, only one of these addresses can be defined as the primary address; the rest are alternative addresses. The primary address is defined during association establishment. The interesting point is that the primary address of an end is determined by the other end. In other words, a source defines the primary address for a destination.

Multistream Delivery One interesting feature of SCTP is the distinction between data transfer and data delivery. SCTP uses TSN numbers to handle data transfer, movement of data chunks between the source and destination. The delivery of the data chunks is controlled by SIs and SSNs. SCTP can support multiple streams, which means that the sender process can define different streams and a message can belong to one of these streams. Each stream is assigned a stream identifier (SI) which uniquely defines that stream.

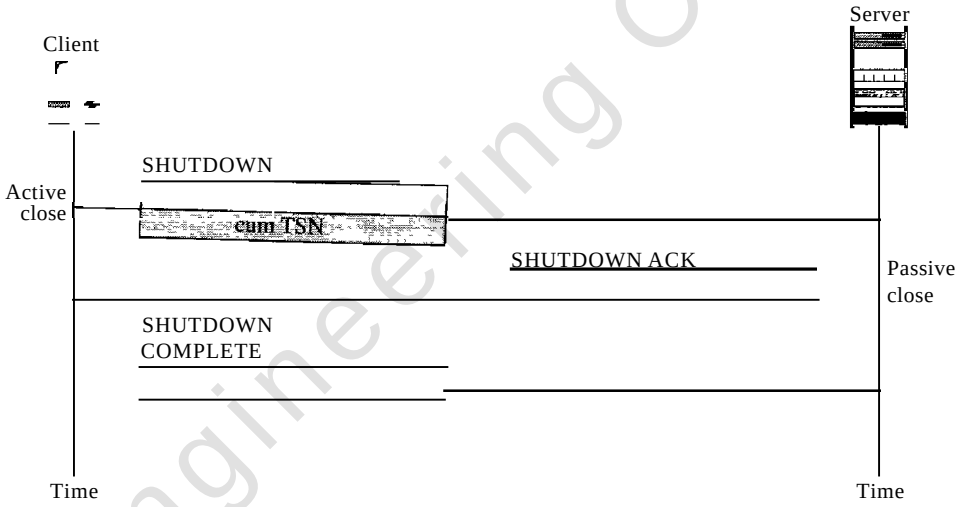
Fragmentation Another issue in data transfer is fragmentation. Although SCTP shares this term with IP, fragmentation in IP and in SCTP belongs to different levels: the former at the network layer, the latter at the transport layer.

SCTP preserves the boundaries of the message from process to process when creating a DATA chunk from a message if the size of the message (when encapsulated in an IP datagram) does not exceed the MTU of the path. The size of an IP datagram carrying a message can be determined by adding the size of the message, in bytes, to the four overheads: data chunk header, necessary SACK chunks, SCTP general header, and IP header. If the total size exceeds the MTU, the message needs to be fragmented.

Association Termination

In SCTP, like TCP, either of the two parties involved in exchanging data (client or server) can close the connection. However, unlike TCP, SCTP does not allow a half-close situation. If one end closes the association, the other end must stop sending new data. If any data are left over in the queue of the recipient of the termination request, they are sent and the association is closed. Association **termination** uses three packets, as shown in Figure 23.35. Note that although the figure shows the case in which termination is initiated by the client, it can also be initiated by the server. Note that there can be several scenarios of association termination. We leave this discussion to the references mentioned at the end of the chapter.

Figure 23.35 *Association termination*



Flow Control

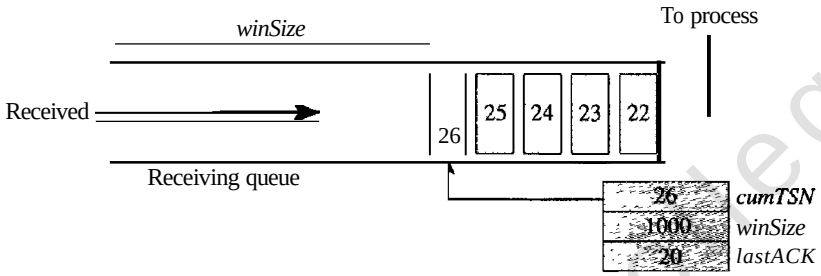
Flow control in SCTP is similar to that in TCP. In TCP, we need to deal with only one unit of data, the byte. In SCTP, we need to handle two units of data, the byte and the chunk. The values of *rwnd* and *cwnd* are expressed in bytes; the values of TSN and acknowledgments are expressed in chunks. To show the concept, we make some unrealistic assumptions. We assume that there is never congestion in the network and that the network is error-free. In other words, we assume that cwnd is infinite and no packet is lost or delayed or arrives out of order. We also assume that data transfer is unidirectional. We correct our unrealistic assumptions in later sections. Current SCTP implementations still use a byte-oriented window for flow control. We, however, show the buffer in terms of chunks to make the concept easier to understand.

Receiver Site

The receiver has one buffer (queue) and three variables. The queue holds the received data chunks that have not yet been read by the process. The first variable holds the last TSN received, *cumTSN*. The second variable holds the available buffer size, *winsize*.

The third variable holds the last accumulative acknowledgment, *lastACK.* Figure 23.36 shows the queue and variables at the receiver site.

Figure 23.36 *Flow control, receiver site*

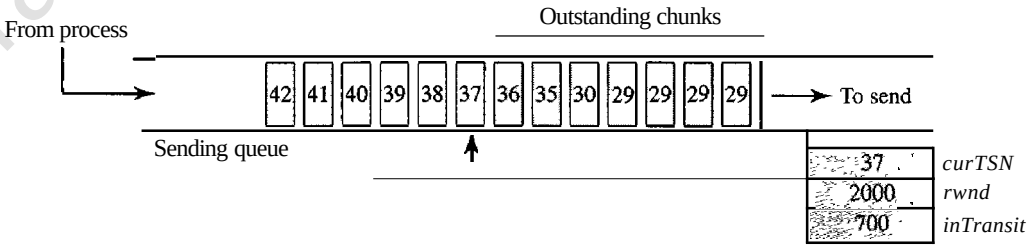


1. When the site receives a data chunk, it stores it at the end of the buffer (queue) and subtracts the size of the chunk from *winSize.* The TSN number of the chunk is stored in the *cumTSN* variable.
2. When the process reads a chunk, it removes it from the queue and adds the size of the removed chunk to *winSize* (recycling).
3. When the receiver decides to send a SACK, it checks the value of *lastAck;* if it is less than *cumTSN,* it sends a SACK with a cumulative TSN number equal to the *cumTSN.* It also includes the value of *winSize* as the advertised window size.

Sender Site

The sender has one buffer (queue) and three variables: *curTSN, rwnd,* and *inTransit,* as shown in Figure 23.37. We assume each chunk is 100 bytes long.

Figure 23.37 *Flow control, sender site*



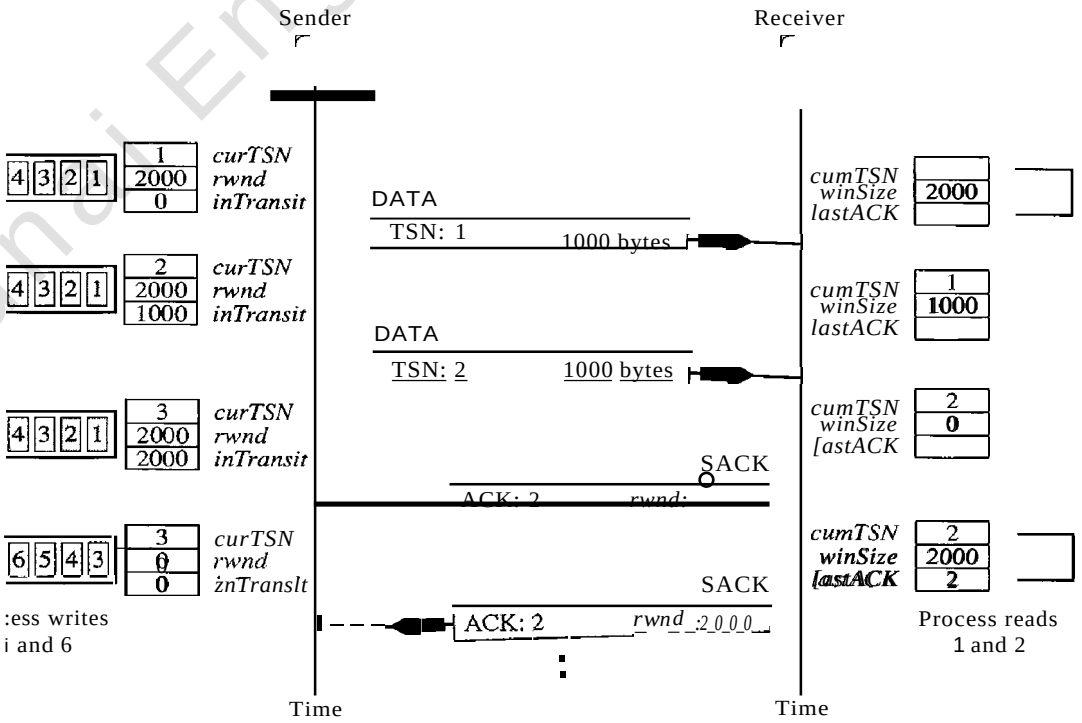
The buffer holds the chunks produced by the process that either have been sent or are ready to be sent. The first variable, *curTSN,* refers to the next chunk to be sent. All chunks in the queue with a TSN less than this value have been sent, but not acknowledged; they are outstanding. The second variable, *rwnd,* holds the last value advertised by the receiver (in bytes). The third variable, *inTransit,* holds the number of bytes in transit, bytes sent but not yet acknowledged. The following is the procedure used by the sender.

1. A chunk pointed to by *curTSN* can be sent if the size of the data is less than or equal to the quantity *rwnd - inTransit.* After sending the chunk, the value of *curTSN* is incremented by 1 and now points to the next chunk to be sent. The value of *inTransit* is incremented by the size of the data in the transmitted chunk.
2. When a SACK is received, the chunks with a TSN less than or equal to the cumulative TSN in the SACK are removed from the queue and discarded. The sender does not have to worry about them any more. The value of *inTransit* is reduced by the total size of the discarded chunks. The value of *rwnd* is updated with the value of the advertised window in the SACK.

A Scenario

Let us give a simple scenario as shown in Figure 23.38. At the start the value of *rwnd* at the sender site and the value of *winSize* at the receiver site are 2000 (advertised during association establishment). Originally, there are four messages in the sender queue. The sender sends one data chunk and adds the number of bytes (1000) to the *inTransit* variable. After awhile, the sender checks the difference between the *rwnd* and *inTransit,* which is 1000 bytes, so it can send another data chunk. Now the difference between the two variables is 0 and no more data chunks can be sent. After awhile, a SACK arrives that acknowledges data chunks 1 and 2. The two chunks are removed from the queue. The value of *inTransit* is now 0. The SACK, however, advertised a receiver window of value 0, which makes the sender.update *rwnd* to 0. Now the sender is blocked; it cannot send any data chunks (with one exception explained later).

Figure 23.38 *Flow control scenario*



At the receiver site, the queue is empty at the beginning. After the first data chunk is received, there is one message in the queue and the value of *cumTSN* is 1. The value of *winSize* is reduced to 1000 because the first message occupies 1000 bytes. After the second data chunk is received, the value of window size is a and *cumTSN* is 2. Now, as we will see, the receiver is required to send a SACK with cumulative TSN of 2. After the first SACK was sent, the process reads the two messages, which means that there is now room in the queue; the receiver advertises the situation with a SACK to allow the sender to send more data chunks. The remaining events are not shown in the figure.

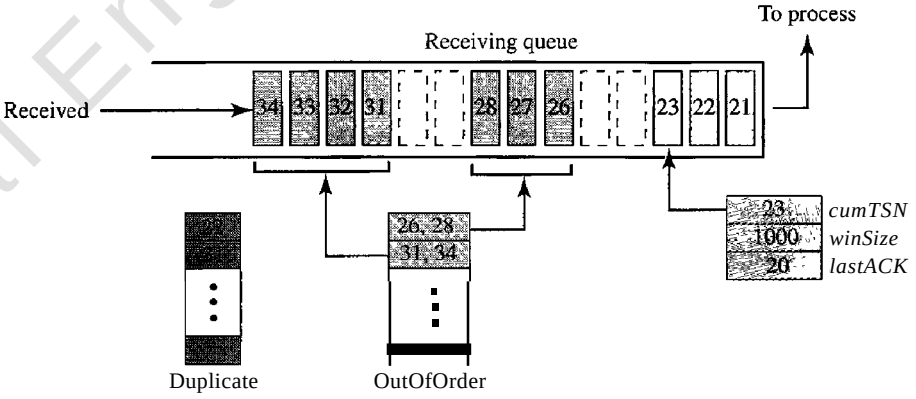
Error Control

SCTP, like TCP, is a reliable transport layer protocol. It uses a SACK chunk to report the state of the receiver buffer to the sender. Each implementation uses a different set of entities and timers for the receiver and sender sites. We use a very simple design to convey the concept to the reader.

Receiver Site

In our design, the receiver stores all chunks that have arrived in its queue including the out-of-order ones. However, it leaves spaces for any missing chunks. It discards duplicate messages, but keeps track of them for reports to the sender. Figure 23.39 shows a typical design for the receiver site and the state of the receiving queue at a particular point in time.

Figure 23.39 *Error control, receiver site*



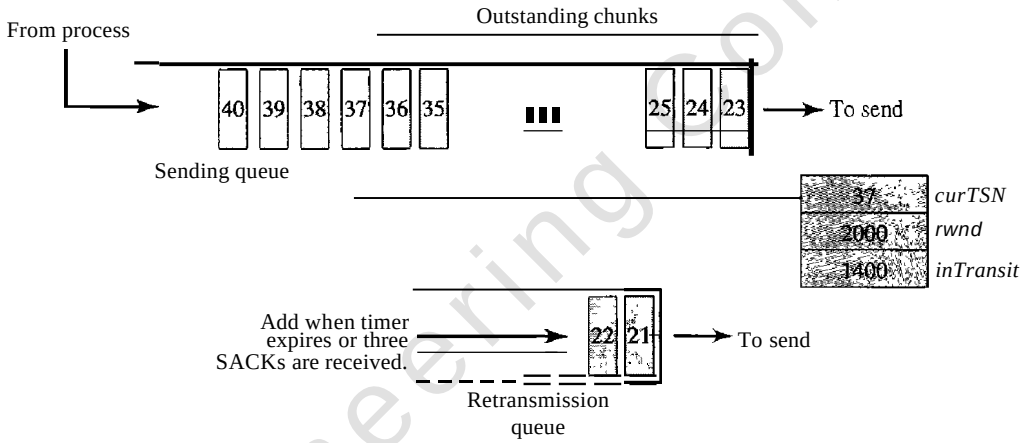
The last acknowledgment sent was for data chunk 20. The available window size is 1000 bytes. Chunks 21 to 23 have been received in order. The first out-of-order block contains chunks 26 to 28. The second out-of-order block contains chunks 31 to 34. A variable holds the value of *cumTSN*. An array of variables keeps track of the beginning and the end of each block that is out of order. An array of variables holds the duplicate chunks received. Note that there is no need for storing duplicate chunks in the queue; they will be discarded. The figure also shows the SACK chunk that will be sent to

report the state of the receiver to the sender. The TSN numbers for out-of-order chunks are relative (offsets) to the cumulative TSN.

Sender Site

At the sender site, our design demands two buffers (queues): a sending queue and a retransmission queue. We also use the three variables *rwnd, inTransit,* and *curTSN* as described in the previous section. Figure 23.40 shows a typical design.

Figure 23.40 *Error control, sender site*



The sending queue holds chunks 23 to 40. The chunks 23 to 36 have already been sent, but not acknowledged; they are outstanding chunks. The *curTSN* points to the next chunk to be sent (37). We assume that each chunk is 100 bytes, which means that 1400 bytes of data (chunks 23 to 36) is in transit. The sender at this moment has a retransmission queue. When a packet is sent, a retransmission timer starts for that packet (all data chunks in that packet). Some implementations use one single timer for the entire association, but we continue with our tradition of one timer for each packet for simplification. When the retransmission timer for a packet expires, or four duplicate SACKs arrive that declare a packet as missing (fast retransmission was discussed in Chapter 12), the chunks in that packet are moved to the retransmission queue to be resent. These chunks are considered lost, rather than outstanding. The chunks in the retransmission queue have priority. In other words, the next time the sender sends a chunk, it would be chunk 21 from the retransmission queue.

Sending Data Chunks

An end can send a data packet whenever there are data chunks in the sending queue with a TSN greater than or equal to *curTSN* or if there are data chunks in the retransmission queue. The retransmission queue has priority. However, the total size of the data chunk or chunks included in the packet must not exceed *rwnd - inTransit,* and the total size of the frame must not exceed the MTU size as we discussed in previous sections.

Retransmission To control a lost or discarded chunk, SCTP, like TCP, employs two strategies: using retransmission timers and receiving four SACKs with the same missing chunks.

Generating SACK Chunks

Another issue in error control is the generation of SACK chunks. The rules for generating SCTP SACK chunks are similar to the rules used for acknowledgment with the TCP ACK flag.

Congestion Control

SCTP, like TCP, is a transport layer protocol with packets subject to congestion in the network. The SCTP designers have used the same strategies we will describe for congestion control in Chapter 24 for TCP. SCTP has slow start (exponential increase), congestion avoidance (additive increase), and congestion detection (multiplicative decrease) phases. Like TCP, SCTP also uses fast retransmission and fast recovery.

23.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

UDP is discussed in Chapter 11 of [For06], Chapter 11 of [Ste94], and Chapter 12 of [ComOO]. TCP is discussed in Chapter 12 of [For06], Chapters 17 to 24 of [Ste94], and Chapter 13 of [ComOO]. SCTP is discussed in Chapter 13 of [For06] and [SX02]. Both UDP and TCP are discussed in Chapter 6 of [Tan03].

Sites

- www.ietf.org/rfc.html Information about RFCs

RFCs

A discussion of UDP can be found in RFC 768.
A discussion of TCP can be found in the following RFCs:

675,700,721,761,793,879,896,1078,1106,1110,1144, 1145, 1146, 1263,1323,1337, 1379,1644,1693,1901,1905,2001,2018,2488,2580

A discussion of SCTP can be found in the following RFCs: .

2960,3257,3284,3285,3286,3309,3436,3554,3708,3758

23.6 KEY TERMS

acknowledgment number
association
association establishment
association termination
byte-oriented
chunk
client
client/server paradigm
connection abortion
connection-oriented service
connectionless service
connectionless, unreliable transport protocol
cookie
COOKIE ACK chunk
COOKIE ECHO chunk
cumulative TSN
DATA chunk
data transfer
denial-of-service attack
dynamic port
ephemeral port number
error control
fast retransmission
flow control
four-way handshaking
fragmentation
full-duplex service
general header
half-close
inbound stream
INIT ACK chunk
INIT chunk
initial sequence number (ISN)
message-oriented
multihoming service
multistream service
port number
primary address
process-to-process delivery
pseudoheader
queue
registered port
retransmission time-out (RTO)
retransmission timer
round-trip time (RTT)
SACK chunk
segment
sequence number
server
simultaneous close
simultaneous open
socket address
stream identifier (SI)
stream sequence number (SSN)
SYN flooding attack
three-way handshaking
Transmission Control Protocol (TCP)
transmission sequence number (TSN)
transport layer
user datagram
User Datagram Protocol (UDP)
verification tag
well-known port number

23.7 SUMMARY

- In the client/server paradigm, an application program on the local host, called the client, needs services from an application program on the remote host, called a server.

- Each application program has a port number that distinguishes it from other programs running at the same time on the same machine.
- The client program is assigned a random port number called an ephemeral port number; the server program is assigned a universal port number called a well-known port number.
- The ICANN has specified ranges for the different types of port numbers.
- The combination of the IP address and the port number, called the socket address, defines a process and a host.
- UDP is a connectionless, unreliable transport layer protocol with no embedded flow or error control mechanism except the checksum for error detection.
- The UDP packet is called a user datagram. A user datagram is encapsulated in the data field of an IP datagram.
- Transmission Control Protocol (TCP) is one of the transport layer protocols in the TCP/IP protocol suite.
- TCP provides process-to-process, full-duplex, and connection-oriented service.
- The unit of data transfer between two devices using TCP software is called a segment; it has 20 to 60 bytes of header, followed by data from the application program.
- A TCP connection normally consists of three phases: connection establishment, data transfer, and connection termination.
- Connection establishment requires three-way handshaking; connection termination requires three- or four-way handshaking.
- TCP uses flow control, implemented as a sliding window mechanism, to avoid overwhelming a receiver with data.
- The TCP window size is determined by the receiver-advertised window size *(rwnd)* or the congestion window size *(cwnd),* whichever is smaller. The window can be opened or closed by the receiver, but should not be shrunk.
- The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.
- TCP uses error control to provide a reliable service. Error control is handled by the checksum, acknowledgment, and time-out. Corrupted and lost segments are retransmitted, and duplicate segments are discarded. Data may arrive out of order and are temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.
- In modem implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.
- SCTP is a message-oriented, reliable protocol that combines the good features of UDP and TCP.
- SCTP provides additional services not provided by UDP or Tep, such as multiple-stream and multihoming services.
- SCTP is a connection-oriented protocol. An SCTP connection is called an association.
- SCTP uses the term *packet* to define a transportation unit.
- In SCTP, control information and data information are carried in separate chunks.

- An SCTP packet can contain control chunks and data chunks with control chunks coming before data chunks.
- In SCTP, each data chunk is numbered using a transmission sequence number (TSN).
- To distinguish between different streams, SCTP uses the sequence identifier (SI).
- To distinguish between different data chunks belonging to the same stream, SCTP uses the stream sequence number (SSN).
- Data chunks are identified by three identifiers: TSN, SI, and SSN. TSN is a cumulative number recognized by the whole association; SSN starts from 0 in each stream.
- SCTP acknowledgment numbers are used only to acknowledge data chunks; control chunks are acknowledged, if needed, by another control chunk.
- An SCTP association is normally established using four packets (four-way handshaking). An association is normally terminated using three packets (three-way handshaking).
- An SCTP association uses a cookie to prevent blind flooding attacks and a verification tag to avoid insertion attacks.
- SCTP provides flow control, error control, and congestion control.
- The SCTP acknowledgment SACK reports the cumulative TSN, the TSN of the last data chunk received in order, and selective TSNs that have been received.

23.8 PRACTICE SET

Review Questions

1. In cases where reliability is not of primary importance, UDP would make a good transport protocol. Give examples of specific cases.
2. Are both UDP and IP unreliable to the same degree? Why or why not?
3. Do port addresses need to be unique? Why or why not? Why are port addresses shorter than IP addresses?
4. What is the dictionary definition of the word *ephemeral?* How does it apply to the concept of the ephemeral port number?
5. What is the minimum size of a UDP datagram?
6. What is the maximum size of a UDP datagram?
7. What is the minimum size of the process data that can be encapsulated in a UDP datagram?
8. What is the maximum size of the process data that can be encapsulated in a UDP datagram?
9. Compare the TCP header and the UDP header. List the fields in the TCP header that are missing from UDP header. Give the reason for their absence.
10. UDP is a message-oriented protocol. TCP is a byte-oriented protocol. If an application needs to protect the boundaries of its message, which protocol should be used, UDP or TCP?

11. What can you say about the TCP segment in which the value of the control field is one of the following?
    a. 000000
    b. 000001
    c. 010001
12. What is the maximum size of the TCP header? What is the minimum size of the TCP header?

Exercises

13. Show the entries for the header of a UDP user datagram that carries a message from a TFTP client to a TFTP server. Fill the checksum field with 0s. Choose an appropriate ephemeral port number and the correct well-known port number. The length of data is 40 bytes. Show the UDP packet, using the format in Figure 23.9.
14. An SNMP client residing on a host with IP address 122.45.12.7 sends a message to an SNMP server residing on a host with IP address 200.112.45.90. What is the pair of sockets used in this communication?
15. A TFTP server residing on a host with IP address 130.45.12.7 sends a message to a TFTP client residing on a host with IP address 14.90.90.33. What is the pair of sockets used in this communication?
16. A client has a packet of 68,000 bytes. Show how this packet can be transferred by using only one UDP user datagram.
17. A client uses UDP to send data to a server. The data are 16 bytes. Calculate the efficiency of this transmission at the UDP level (ratio of useful bytes to total bytes).
18. Redo Exercise 17, calculating the efficiency of transmission at the IP leveL Assume no options for the IP header.
19. Redo Exercise 18, calculating the efficiency of transmission at the data link layer. Assume no options for the IP header and use Ethernet at the data link layer.
20. The following is a dump of a UDP header in hexadecimal format.

    0632000DOO lCE217

    a. What is the source port number?
    b. What is the destination port number?
    c. What is the total length of the user datagram?
    d. What is the length of the data?
    e. Is the packet directed from a client to a server or vice versa?
    f. What is the client process?
21. An IP datagram is carrying a TCP segment destined for address 130.14.16.17/16. The destination port address is corrupted, and it arrives at destination 130.14.16.19/16. How does the receiving TCP react to this error?
22. In TCP, if the value of HLEN is 0111, how many bytes of option are included in the segment?

23. Show the entries for the header of a TCP segment that carries a message from an FTP client to an FTP server. Fill the checksum field with Os. Choose an appropriate ephemeral port number and the correct well-known port number. The length of the data is 40 bytes.
24. The following is a dump of a TCP header in hexadecimal format.

    05320017 00000001 00000000 5OO207FF 00000000

    a. What is the source port number?
    b. What is the destination port number?
    c. What the sequence number?
    d. What is the acknowledgment number?
    e. What is the length of the header?
    f. What is the type of the segment?
    g. What is the window size?
25. To make the initial sequence number a random number, most systems start the counter at 1 during bootstrap and increment the counter by 64,000 every 0.5 s. How long does it take for the counter to wrap around?
26. In a connection, the value of *cwnd* is 3000 and the value of *rwnd* is 5000. The host has sent 2000 bytes which has not been acknowledged. How many more bytes can be sent?
27. TCP opens a connection using an initial sequence number (ISN) of 14,534. The other party opens the connection with an ISN of 21,732. Show the three TCP segments during the connection establishment.
28. A client uses TCP to send data to a server. The data are 16 bytes. Calculate the efficiency of this transmission at the TCP level (ratio of useful bytes to total bytes). Calculate the efficiency of transmission at the IP level. Assume no options for the IP header. Calculate the efficiency of transmission at the data link layer. Assume no options for the IF header and use Ethernet at the data link layer.
29. TCP is sending data at 1 Mbyte/s. If the sequence number starts with 7000, how long does it take before the sequence number goes back to zero?
30. A TCP connection is using a window size of 10,000 bytes, and the previous acknowledgment number was 22,001. It receives a segment with acknowledgment number 24,001 and window size advertisement of 12,000. Draw a diagram to show the situation of the window before and after.
31. A window holds bytes 2001 to 5000. The next byte to be sent is 3001. Draw a figure to show the situation of the window after the following two events.
    a. An ACK segment with the acknowledgment number 2500 and window size advertisement 4000 is received.
    b. A segment carrying 1000 bytes is sent.
32. In SCTP, the value of the cumulative TSN in a SACK is 23. The value of the previous cumulative TSN in the SACK was 29. What is the problem?
33. In SCTP, the state of a receiver is as follows:
    a. The receiving queue has chunks 1 to 8, 11 to 14, and 16 to 20.
    b. There are 1800 bytes of space in the queue.

c. The value of *lastAck* is 4.
d. No duplicate chunk has been received.
e. The value of *cumTSN* is 5.

Show the contents of the receiving queue and the variables.

34. In SCTP, the state of a sender is as follows:
    a. The sending queue has chunks 18 to 23.
    b. The value of *cumTSN* is 20.
    c. The value of the window size is 2000 bytes.
    d. The value of *inTransit* is 200.

    If each data chunk contains 100 bytes of data, how many DATA chunks can be sent now? What is the next DATA chunk to be sent?

Research Activities

35. Find more information about ICANN. What was it called before its name was changed?
36. TCP uses a transition state diagram to handle sending and receiving segments. Find out about this diagram and how it handles flow and control.
37. SCTP uses a transition state diagram to handle sending and receiving segments. Find out about this diagram and how it handles flow and control.
38. What is the half-open case in TCP?
39. What is the half-duplex close case in TCP?
40. The *tcpdump* command in UNIX or LINUX can be used to print the headers of packets of a network interface. Use *tcpdump* to see the segments sent and received.
41. In SCTP, find out what happens if a SACK chunk is delayed or lost.
42. Find the name and functions of timers used in TCP.
43. Find the name and functions of timers used in SCTP.
44. Find out more about ECN in SCTP. Find the format of these two chunks.
45. Some application programs, such as FTP, need more than one connection when using TCP. Find how the multistream service of SCTP can help these applications establish only one association with several streams.

CHAPTER 24

Congestion Control and Quality of Service

Congestion control and quality of service are two issues so closely bound together that improving one means improving the other and ignoring one usually means ignoring the other. Most techniques to prevent or eliminate congestion also improve the quality of service in a network.

We have postponed the discussion of these issues until now because these are issues related not to one layer, but to three: the data link layer, the network layer, and the transport layer. We waited until now so that we can discuss these issues once instead of repeating the subject three times. Throughout the chapter, we give examples of congestion control and quality of service at different layers.

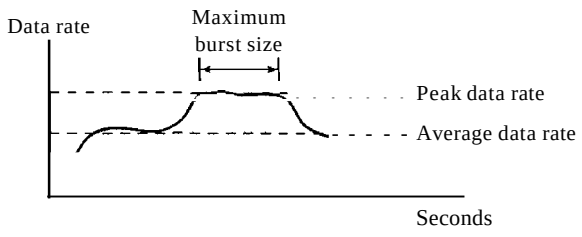
24.1 DATA TRAFFIC

The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic. So, before talking about congestion control and quality of service, we discuss the data traffic itself.

Traffic Descriptor

Traffic descriptors are qualitative values that represent a data flow. Figure 24.1 shows a traffic flow with some of these values.

Figure 24.1 *Traffic descriptors*



Average Data Rate

The average data rate is the number of bits sent during a period of time, divided by the number of seconds in that period. We use the following equation:

$$\text{Average data rate} = \frac{\text{amount of data}}{\text{time}}$$

The average data rate is a very useful characteristic of traffic because it indicates the average bandwidth needed by the traffic.

Peak Data Rate

The peak data rate defines the maximum data rate of the traffic. In Figure 24.1 it is the maximum *y* axis value. The peak data rate is a very important measurement because it indicates the peak bandwidth that the network needs for traffic to pass through without changing its data flow.

Maximum Burst Size

Although the peak data rate is a critical value for the network, it can usually be ignored if the duration of the peak value is very short. For example, if data are flowing steadily at the rate of 1 Mbps with a sudden peak data rate of 2 Mbps for just 1 ms, the network probably can handle the situation. However, if the peak data rate lasts 60 ms, there may be a problem for the network. The maximum burst size normally refers to the maximum length of time the traffic is generated at the peak rate.

Effective Bandwidth

The effective bandwidth is the bandwidth that the network needs to allocate for the flow of traffic. The effective bandwidth is a function of three values: average data rate, peak data rate, and maximum burst size. The calculation of this value is very complex.

Traffic Profiles

For our purposes, a data flow can have one of the following traffic profiles: constant bit rate, variable bit rate, or bursty as shown in Figure 24.2.

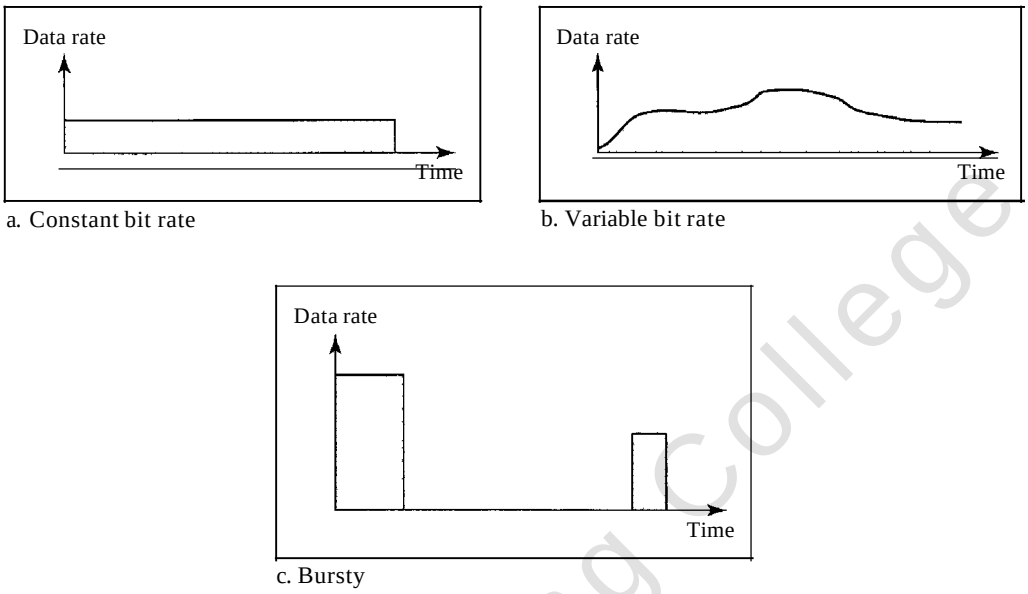
Constant Bit Rate

A constant-bit-rate (CBR), or a fixed-rate, traffic model has a data rate that does not change. In this type of flow, the average data ratc and thc peak data rate are the same. The maximum burst size is not applicable. This type of traffic is very easy for a network to handle since it is predictable. The network knows in advance how much bandwidth to allocate for this type of flow.

Variable Bit Rate

In the variable-bit-rate (VBR) category, the rate of the data flow changes in time, with the changes smooth instead of sudden and sharp. In this type of flow, the average data

Figure 24.2 *Three traffic profiles*



rate and the peak data rate are different. The maximum burst size is usually a small value. This type of traffic is more difficult to handle than constant-bit-rate traffic, but it normally does not need to be reshaped, as we will see later.

Bursty

In the **bursty data** category, the data rate changes suddenly in a very short time. It may jump from zero, for example, to 1 Mbps in a few microseconds and vice versa. It may also remain at this value for a while. The average bit rate and the peak bit rate are very different values in this type of flow. The maximum burst size is significant. This is the most difficult type of traffic for a network to handle because the profile is very unpredictable. To handle this type of traffic, the network normally needs to reshape it, using reshaping techniques, as we will see shortly. Bursty traffic is one of the main causes of congestion in a network.

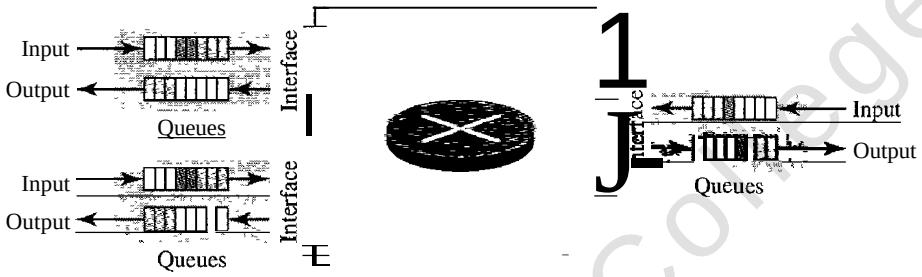
24.2 CONGESTION

An important issue in a packet-switched network is **congestion.** Congestion in a network may occur if the **load** on the network-the number of packets sent to the network-is greater than the *capacity* of the network-the number of packets a network can handle. **Congestion control** refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

We may ask why there is congestion on a network. Congestion happens in any system that involves waiting. For example, congestion happens on a freeway because any abnonnality in the flow, such as an accident during rush hour, creates blockage.

Congestion in a network or internetwork occurs because routers and switches have queues-buffers that hold the packets before and after processing. A router, for example, has an input queue and an output queue for each interface. When a packet arrives at the incoming interface, it undergoes three steps before departing, as shown in Figure 24.3.

Figure 24.3 *Queues in a router*



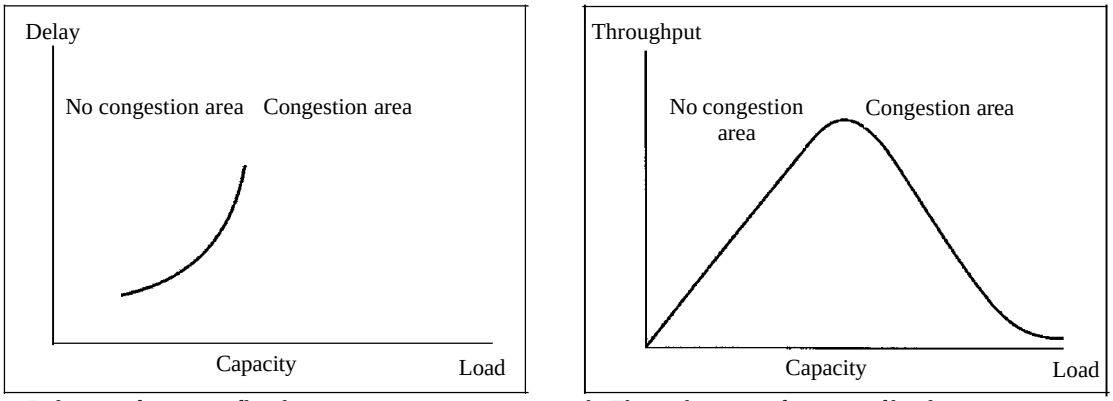
1. The packet is put at the end of the input queue while waiting to be checked.
2. The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its routing table and the destination address to find the route.
3. The packet is put in the appropriate output queue and waits its tum to be sent.

We need to be aware of two issues. First, if the rate of packet arrival is higher than the packet processing rate, the input queues become longer and longer. Second, if the packet departure rate is less than the packet processing rate, the output queues become longer and longer.

Network Performance

Congestion control involves two factors that measure the performance of a network: *delay* and *throughput.* Figure 24.4 shows these two performance measures as function of load.

Figure 24.4 *Packet delay and throughput as functions of load*



a. Delay as a function of load

b. Throughput as a function of load

Delay Versus Load

Note that when the load is much less than the capacity of the network, the delay is at a minimum. This minimum delay is composed of propagation delay and processing delay, both of which are negligible. However, when the load reaches the network capacity, the delay increases sharply because we now need to add the waiting time in the queues (for all routers in the path) to the total delay. Note that the delay becomes infinite when the load is greater than the capacity. If this is not obvious, consider the size of the queues when almost no packet reaches the destination, or reaches the destination with infinite delay; the queues become longer and longer. Delay has a negative effect on the load and consequently the congestion. When a packet is delayed, the source, not receiving the acknowledgment, retransmits the packet, which makes the delay, and the congestion, worse.

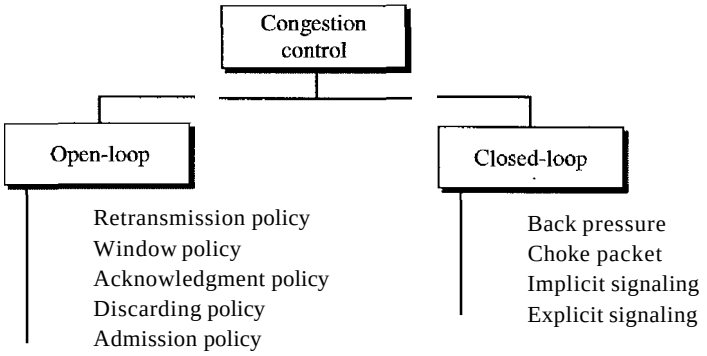
Throughput Versus Load

We defined throughput in Chapter 3 as the number of bits passing through a point in a second. We can extend that definition from bits to packets and from a point to a network. We can define throughput in a network as the number of packets passing through the network in a unit of time. Notice that when the load is below the capacity of the network, the throughput increases proportionally with the *load.* We expect the throughput to remain constant after the load reaches the capacity, but instead the throughput declines sharply. The reason is the discarding of packets by the routers. When the load exceeds the capacity, the queues become full and the routers have to discard some packets. Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets, using time-out mechanisms, when the packets do not reach the destinations.

24.3 CONGESTION CONTROL

Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal) as shown in Figure 24.5.

Figure 24.5 *Congestion control categories*



Open-Loop Congestion Control

In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination. We give a brief list of policies that can prevent congestion.

Retransmission Policy

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network. However, a good retransmission policy can prevent congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion. For example, the retransmission policy used by TCP (explained later) is designed to prevent or alleviate congestion.

Window Policy

The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control. In the *Go-Back-N* window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

Acknowledgment Policy

The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion. Several approaches are used in this case. A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires. A receiver may decide to acknowledge only *N* packets at a time. We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

Discarding Policy

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Admission Policy

An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

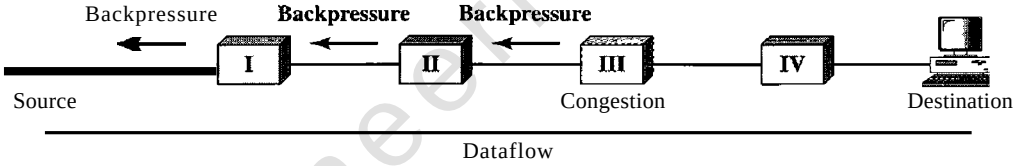
Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols. We describe a few of them here.

Backpressure

The technique of *backpressure* refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes. And so on. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is corning. Figure 24.6 shows the idea of backpressure.

Figure 24.6 *Backpressure method for alleviating congestion*



Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I informs the source of data to slow down. This, in time, alleviates the congestion. Note that the *pressure* on node III is moved backward to the source to remove the congestion.

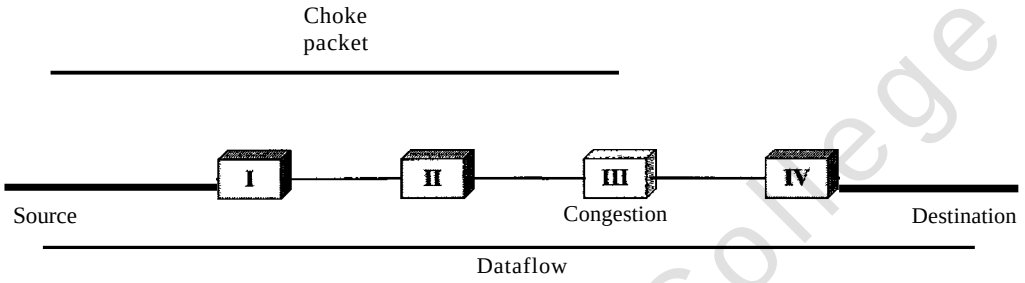
None of the virtual-circuit networks we studied in this book use backpressure. It was, however, implemented in the first virtual-circuit network, X.25. The technique cannot be implemented in a datagram network because in this type of network, a node (router) does not have the slightest knowledge of the upstream router.

Choke Packet

A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has traveled are not warned. We have seen an example of this type of control in ICMP. When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them; but it informs the source

host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers, and does not take any action. Figure 24.7 shows the idea of a choke packet.

Figure 24.7 *Choke packet*



Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms. For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. We will see this type of signaling when we discuss TCP congestion control later in the chapter.

Explicit Signaling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method, however, is different from the choke packet method. In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data. Explicit signaling, as we will see in Frame Relay congestion control, can occur in either the forward or the backward direction.

Backward Signaling A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

Forward Signaling A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

24.4 TWO EXAMPLES

To better understand the concept of congestion control, let us give two examples: one in TCP and the other in Frame Relay.

Congestion Control in TCP

We discussed TCP in Chapter 23. We now show how TCP uses congestion control to avoid congestion or alleviate congestion in the network.

Congestion Window

In Chapter 23, we talked about flow control and tried to discuss solutions when the receiver is overwhelmed with data. We said that the sender window size is determined by the available buffer space in the receiver *(rwnd).* In other words, we assumed that it is only the receiver that can dictate to the sender the size of the sender's window. We totally ignored another entity here-the network. If the network cannot deliver the data as fast as they are created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Today, the sender's window size is determined not only by the receiver but also by congestion in the network.

The sender has two pieces of information: the receiver-advertised window size and the congestion window size. The actual size of the window is the minimum of these two.

$$\text{Actual window size} = \text{minimum (rwnd, cwnd)}$$

We show shortly how the size of the congestion window *(cwnd)* is determined.

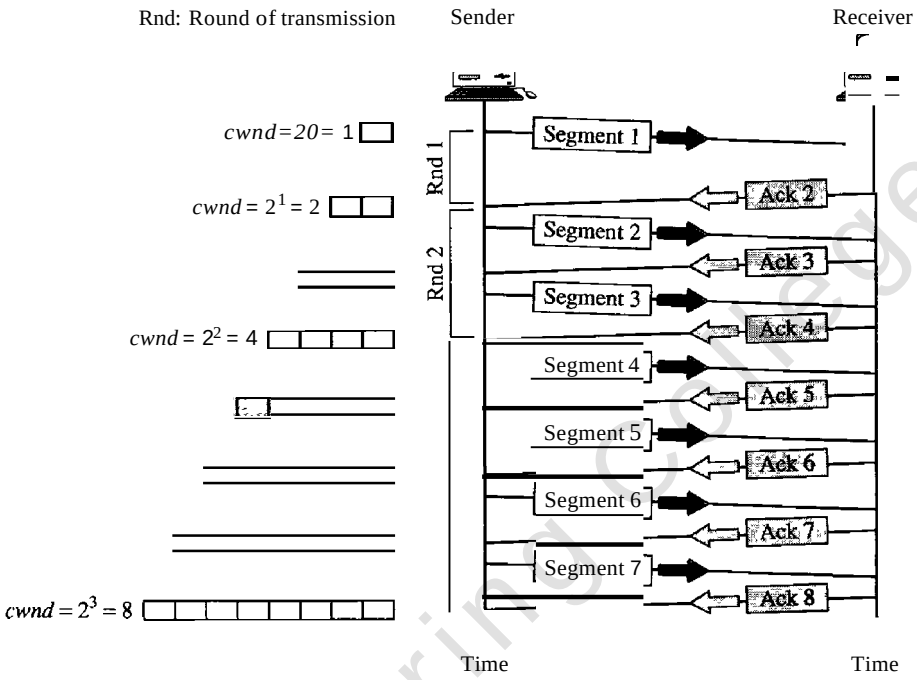
Congestion Policy

TCP's general policy for handling congestion is based on three phases: slow start, congestion avoidance, and congestion detection. In the slow-start phase, the sender starts with a very slow rate of transmission, but increases the rate rapidly to reach a threshold. When the threshold is reached, the data rate is reduced to avoid congestion. Finally if congestion is detected, the sender goes back to the slow-start or congestion avoidance phase based on how the congestion is detected.

Slow Start: Exponential Increase One of the algorithms used in TCP congestion control is called slow start. This algorithm is based on the idea that the size of the congestion window *(cwnd)* starts with one maximum segment size (MSS). The MSS is determined during connection establishment by using an option of the same name. The size of the window increases one MSS each time an acknowledgment is received. As the name implies, the window starts slowly, but grows exponentially. To show the idea, let us look at Figure 24.8. Note that we have used three simplifications to make the discussion more understandable. We have used segment numbers instead of byte numbers (as though each segment contains only 1 byte). We have assumed that *rwnd* is much higher than *cwnd,* so that the sender window size always equals *cwnd.* We have assumed that each segment is acknowledged individually.

The sender starts with $cwnd = 1$ MSS. This means that the sender can send only one segment. After receipt of the acknowledgment for segment 1, the size of the congestion window is increased by 1, which means that *cwnd* is now 2. Now two more segments can be sent. When each acknowledgment is received, the size of the window is increased by 1 MSS. When all seven segments are acknowledged, $cwnd = 8$.

Figure 24.8 *Slow start, exponential increase*



If we look at the size of *cwnd* in terms of rounds (acknowledgment of the whole window of segments), we find that the rate is exponential as shown below:

Start	➡	$cwnd = 1$
After round 1	➡	$cwnd = 2^1 = 2$
After round 2	➡	$cwnd = 2^2 = 4$
After round 3	➡	$cwnd = 2^3 = 8$

We need to mention that if there is delayed ACKs, the increase in the size of the window is less than power of 2.

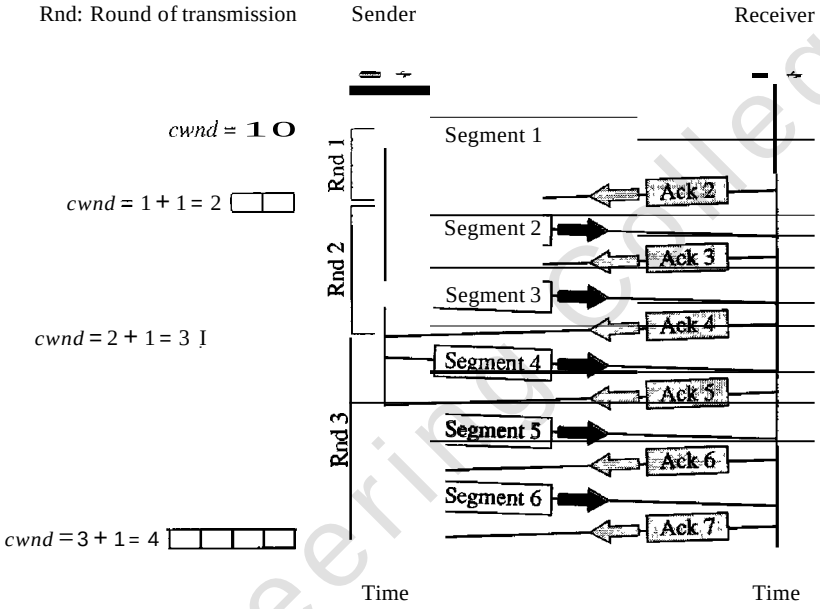
Slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named *ssthresh* (slow-start threshold). When the size of window in bytes reaches this threshold, slow start stops and the next phase starts. In most implementations the value of *ssthresh* is 65,535 bytes.

In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

Congestion Avoidance: Additive Increase If we start with the slow-start algorithm, the size of the congestion window increases exponentially. To avoid congestion before it happens, one must slow down this exponential growth. TCP defines another algorithm called congestion avoidance, which undergoes an additive increase instead of an exponential one. When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole window of segments is acknowledged (one round), the size of the

congestion window is increased by 1. To show the idea, we apply this algorithm to the same scenario as slow start, although we will see that the congestion avoidance algorithm usually starts when the size of the window is much greater than 1. Figure 24.9 shows the idea.

Figure 24.9 *Congestion avoidance, additive increase*



In this case, after the sender has received acknowledgments for a complete window size of segments, the size of the window is increased by one segment.

If we look at the size of *cwnd* in terms of rounds, we find that the rate is additive as shown below:

Start	➡	*cwnd*=*l*
After round 1	➡	*cwnd*= 1+1=2
After round 2	➡	*cwnd*=*2*+ 1=3
After round 3	➡	*cwnd*=*3*+ 1=4

In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

Congestion Detection: Multiplicative Decrease If congestion occurs, the congestion window size must be decreased. The only way the sender can guess that congestion has occurred is by the need to retransmit a segment. However, retransmission can occur in one of two cases: when a timer times out or when three ACKs are received. In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease. Most TCP implementations have two reactions:

1. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments.

In this case TCP reacts strongly:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets *cwnd* to the size of one segment.
c. It starts the slow-start phase again.

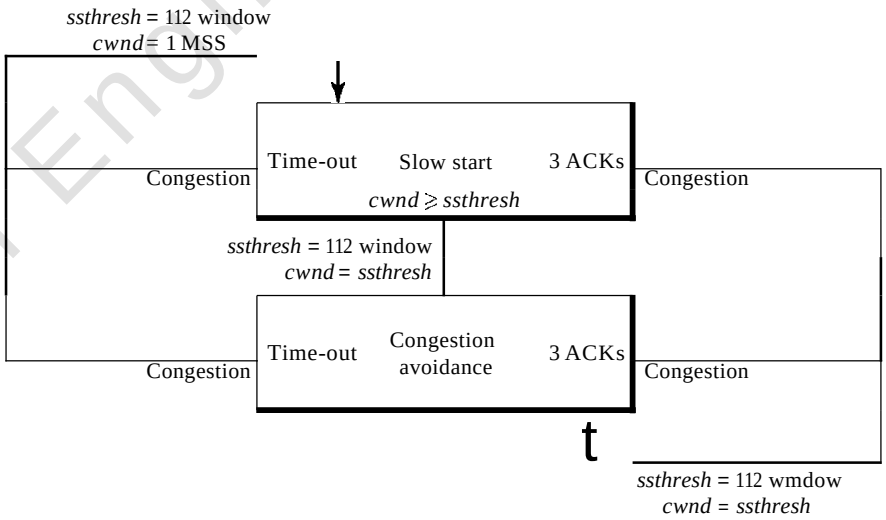
2. If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction:
   a. It sets the value of the threshold to one-half of the current window size.
   b. It sets *cwnd* to the value of the threshold (some implementations add three segment sizes to the threshold).
   c. It starts the congestion avoidance phase.

An implementations reacts to congestion detection in one of the following ways:

- If detection is by time-out, a new *slow-start* phase starts.
- If detection is by three ACKs, a new *congestion avoidance* phase starts.

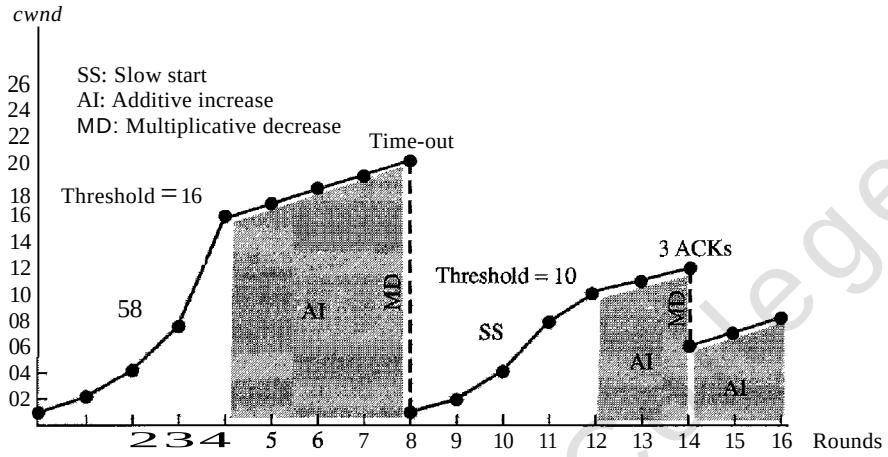
Summary In Figure 24.10, we summarize the congestion policy of TCP and the relationships between the three phases.

Figure 24.10 *TCP congestion policy summary*



We give an example in Figure 24.11. We assume that the maximum window size is 32 segments. The threshold is set to 16 segments (one-half of the maximum window size). In the *slow-start* phase the window size starts from 1 and grows exponentially until it reaches the threshold. After it reaches the threshold, the *congestion avoidance (additive increase)* procedure allows the window size to increase linearly until a time-out occurs or the maximum window size is reached. In Figure 24.11, the time-out occurs when the window size is 20. At this moment, the *multiplicative decrease* procedure takes

Figure 24.11 *Congestion example*



over and reduces the threshold to one-half of the previous window size. The previous window size was 20 when the time-out happened so the new threshold is now 10.

TCP moves to slow start again and starts with a window size of 1, and TCP moves to additive increase when the new threshold is reached. When the window size is 12, a three-ACKs event happens. The multiplicative decrease procedure takes over again. The threshold is set to 6 and TCP goes to the additive increase phase this time. It remains in this phase until another time-out or another three ACKs happen.

Congestion Control in Frame Relay

Congestion in a Frame Relay network decreases throughput and increases delay. A high throughput and low delay are the main goals of the Frame Relay protocol. Frame Relay does not have flow control. In addition, Frame Relay allows the user to transmit bursty data. This means that a Frame Relay network has the potential to be really congested with traffic, thus requiring congestion control.

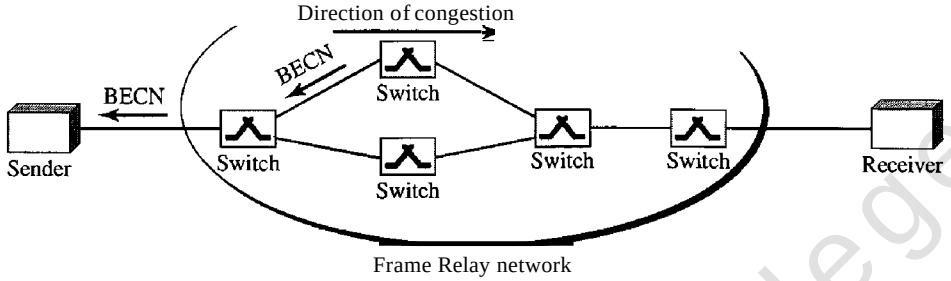
Congestion Avoidance

For congestion avoidance, the Frame Relay protocol uses 2 bits in the frame to explicitly warn the source and the destination of the presence of congestion.

BECN The backward explicit congestion notification (BECN) bit warns the sender of congestion in the network. One might ask how this is accomplished since the frames are traveling away from the sender. In fact, there are two methods: The switch can use response frames from the receiver (full-duplex mode), or else the switch can use a predefined connection (DLCI = 1023) to send special frames for this specific purpose. The sender can respond to this warning by simply reducing the data rate. Figure 24.12 shows the use of BECN.

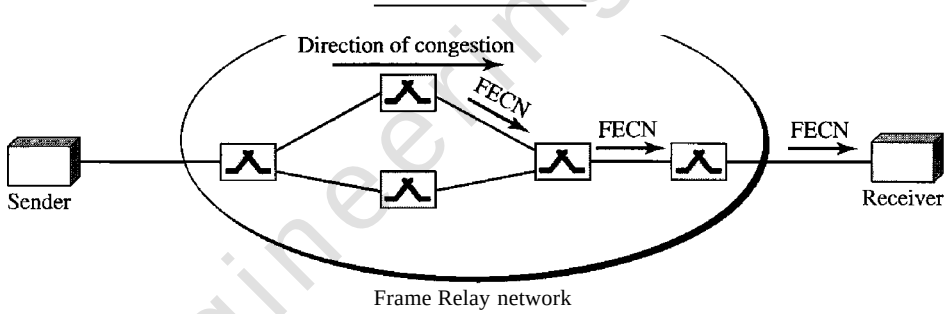
FECN The forward explicit congestion notification (FECN) bit is used to warn the receiver of congestion in the network. It might appear that the receiver cannot do anything to relieve the congestion. However, the Frame Relay protocol assumes that the sender and receiver are communicating with each other and are using some type of flow control at a higher level. For example, if there is an acknowledgment mechanism at this

Figure 24.12 *BECN*



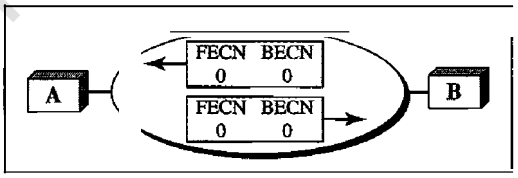
higher level, the receiver can delay the acknowledgment, thus forcing the sender to slow down. Figure 24.13 shows the use ofFECN.

Figure 24.13 *FECN*

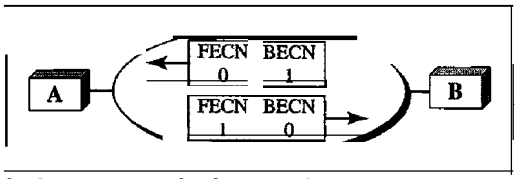


When two endpoints are communicating using a Frame Relay network, four situations may occur with regard to congestion. Figure 24.14 shows these four situations and the values of FECN and BECN.

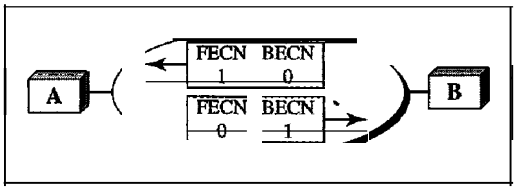
Figure 24.14 *Four cases ofcongestion*



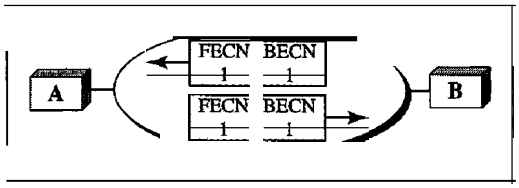
a. No congestion



b. Congestion in the direction A-B



c. Congestion in the direction B-A



d. Congestion in both directions

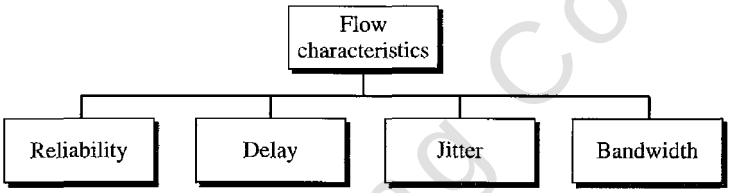
24.5 QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.

Flow Characteristics

Traditionally, four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth, as shown in Figure 24.15.

Figure 24.15 *Flow characteristics*



Reliability

Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. However, the sensitivity of application programs to reliability is not the same. For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay

Source-to-destination delay is another flow characteristic. Again applications can tolerate delay in different degrees. In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

Jitter

Jitter is the variation in delay for packets belonging to the same flow. For example, if four packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time. On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21,22, 19, and 24.

For applications such as audio and video, the first case is completely acceptable; the second case is not. For these applications, it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets. For this application, the second case is not acceptable.

Jitter is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.

In Chapter 29, we show how multimedia communication deals with jitter. If the jitter is high, some action is needed in order to use the received data.

Bandwidth

Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

Flow Classes

Based on the flow characteristics, we can classify flows into groups, with each group having similar levels of characteristics. This categorization is not formal or universal; some protocols such as ATM have defined classes, as we will see later.

24.6 lECHNIQUES TO IMPROVE QoS

In Section 24.5 we tried to define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.

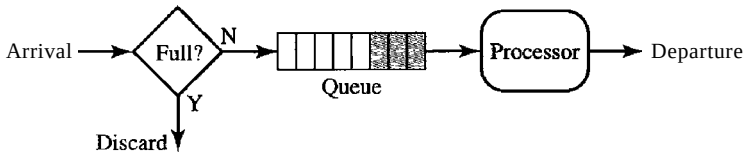
Scheduling

Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. We discuss three of them here: FIFO queuing, priority queuing, and weighted fair queuing.

FIFO Queuing

In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop. Figure 24.16 shows a conceptual view of a FIFO queue.

Figure 24.16 *FIFO queue*

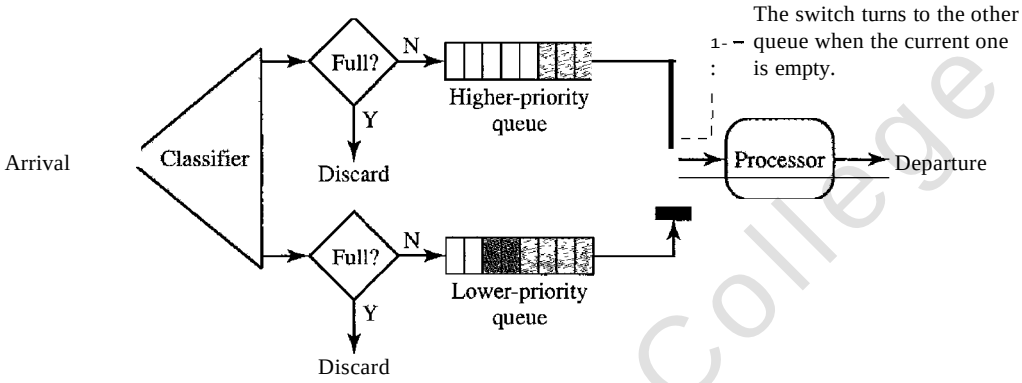


Priority Queuing

In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving

a queue until it is empty. Figure 24.17 shows priority queuing with two priority levels (for simplicity).

Figure 24.17 *Priority queuing*



A priority queue can provide better QoS than the FIFO queue because higher-priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called *starvation.*

Weighted Fair Queuing

A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight. For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equaL In this way, we have fair queuing with priority. Figure 24.18 shows the technique with three classes.

Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Figure 24.19 shows a leaky bucket and its effects.

Figure 24.18 *Weighted fair queuing*

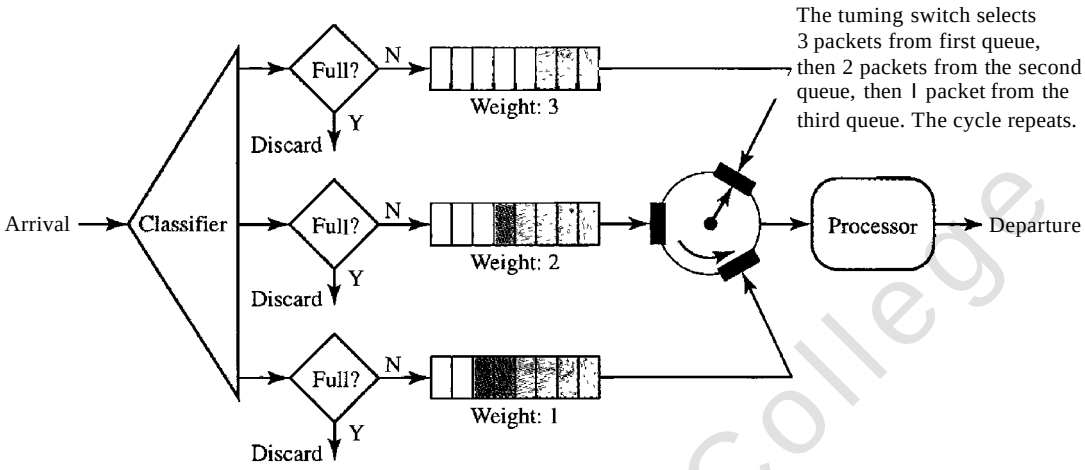
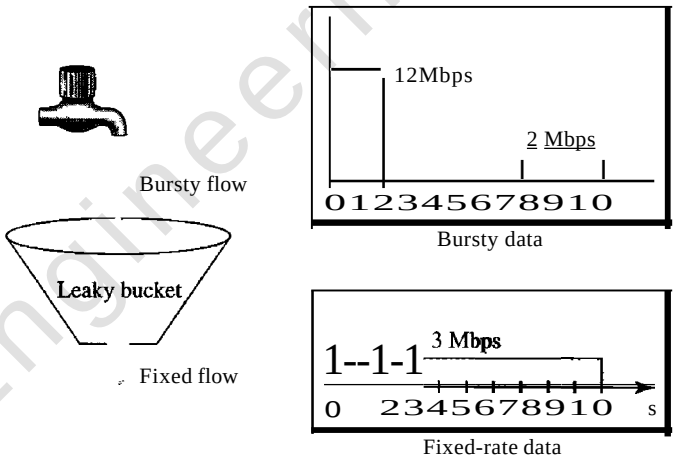


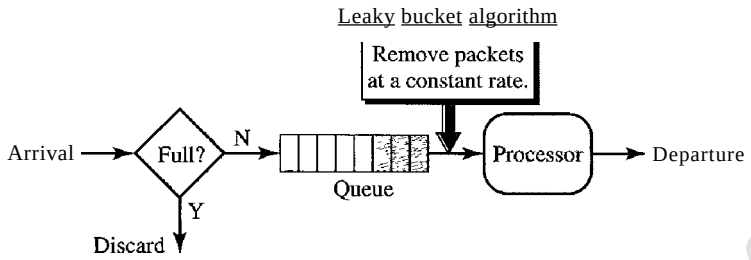
Figure 24.19 *Leaky bucket*



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 24.19 the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in lOs. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion. As an analogy, consider the freeway during rush hour (bursty traffic). If, instead, commuters could stagger their working hours, congestion o'n our freeways could be avoided.

A simple leaky bucket implementation is shown in Figure 24.20. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM

Figure 24.20 *Leaky bucket implementation*



networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to *n* at the tick of the clock.
2. If *n* is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until *n* is smaller than the packet size.
3. Reset the counter and go to step 1.

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

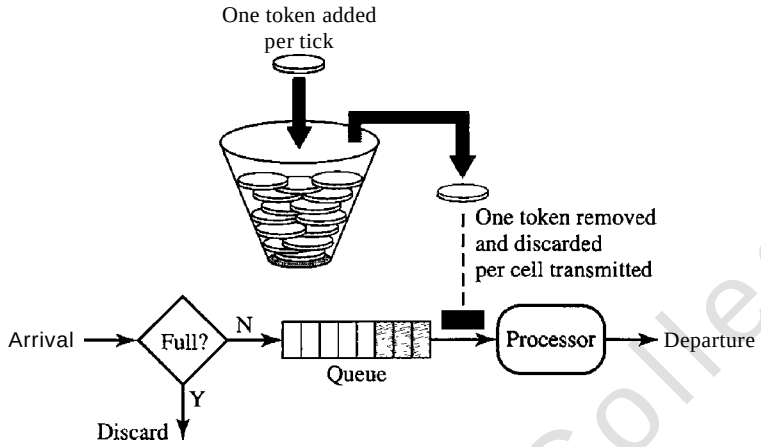
Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends *n* tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if *n* is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. Figure 24.21 shows the idea.

The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

The token bucket allows bursty traffic at a regulated maximum rate.

Figure 24.21 *Token bucket*



Combining Token Bucket and Leaky Bucket

The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.

Resource Reservation

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand. We discuss in this section one QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

Admission Control

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

24.7 INTEGRATED SERVICES

Based on the topics in Sections 24.5 and 24.6, two models have been designed to provide quality of service in the Internet: Integrated Services and Differentiated Services. Both models emphasize the use of quality of service at the network layer (IP), although the model can also be used in other layers such as the data link. We discuss Integrated Services in this section and Differentiated Service in Section 24.8.

As we learned in Chapter 20, IP was originally designed for *best-effort* delivery. This means that every user receives the same level of services. This type of delivery

does not guarantee the minimum of a service, such as bandwidth, to applications such as real-time audio and video. If such an application accidentally gets extra bandwidth, it may be detrimental to other applications, resulting in congestion.

Integrated Services, sometimes called IntServ, is *afiow-based* QoS model, which means that a user needs to create a flow, a kind of virtual circuit, from the source to the destination and inform all routers of the resource requirement.

Integrated Services is a *flow-based* QoS model designed for IP.

Signaling

The reader may remember that IP is a connectionless, datagram, packet-switching protocol. How can we implement a flow-based model over a connectionless protocol? The solution is a signaling protocol to run over IP that provides the signaling mechanism for making a reservation. This protocol is called Resource Reservation Protocol (RSVP) and will be discussed short!y.

Flow Specification

When a source makes a reservation, it needs to define a flow specification. A flow spedfication has two parts: Rspec (resource specification) and Tspec (traffic specification). Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.). Tspec defines the traffic characterization of the flow.

Admission

After a router receives the flow specification from an application, it decides to admit or deny the service. The decision is based on the previous commitments of the router and the current availability of the resource.

Service Classes

Two classes of services have been defined for Integrated Services: guaranteed service and controlled-load service.

Guaranteed Service Class

This type of service is designed for real-time traffic that needs a guaranteed minimum end-to-end delay. The end-to-end delay is the sum of the delays in the routers, the propagation delay in the media, and the setup mechanism. Only the first, the sum of the delays in the routers, can be guaranteed by the router. This type of service guarantees that the packets will arrive within a certain delivery time and are not discarded if flow traffic stays within the boundary of Tspec. We can say that guaranteed services are quantitative services, in which the amount of end-to-end delay and the data rate must be defined by the application.

Controlled-Load Service Class

This type of service is designed for applications that can accept some delays, but are sensitive to an overloaded network and to the danger of losing packets. Good examples of these types of applications are file transfer, e-mail, and Internet access. The controlled-load service is a qualitative type of service in that the application requests the possibility of low-loss or no-loss packets.

RSVP

In the Integrated Services model, an application program needs resource reservation. As we learned in the discussion of the IntServ model, the resource reservation is for a *flow.* This means that if we want to use IntServ at the IP level, we need to create a flow, a kind of virtual-circuit network, out of the IP, which was originally designed as a datagram packet-switched network. A virtual-circuit network needs a signaling system to set up the virtual circuit before data traffic can start. The Resource Reservation Protocol (RSVP) is a signaling protocol to help IP create a flow and consequently make a resource reservation. Before discussing RSVP, we need to mention that it is an independent protocol separate from the Integrated Services model. It may be used in other models in the future.

Multicast Trees

RSVP is different from some other signaling systems we have seen before in that it is a signaling system designed for multicasting. However, RSVP can be also used for unicasting because unicasting is just a special case of multicasting with only one member in the multicast group. The reason for this design is to enable RSVP to provide resource reservations for all kinds of traffic including multimedia which often uses multicasting.

Receiver-Based Reservation

In RSVP, the receivers, not the sender, make the reservation. This strategy matches the other multicasting protocols. For example, in multicast routing protocols, the receivers, not the sender, make a decision to join or leave a multicast group.

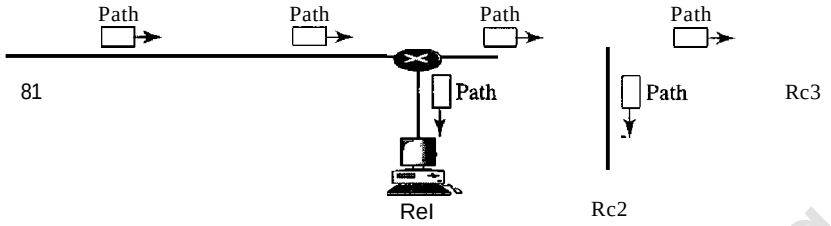
RSVP Messages

RSVP has several types of messages. However, for our purposes, we discuss only two of them: **Path** and Resv.

Path Messages Recall that the receivers in a flow make the reservation in RSVP. However, the receivers do not know the path traveled by packets before the reservation is made. The path is needed for the reservation. To solve the problem, RSVP uses *Path* messages. A Path message travels from the sender and reaches all receivers in the multicast path. On the way, a Path message stores the necessary information for the receivers. A Path message is sent in a multicast environment; a new message is created when the path diverges. Figure 24.22 shows path messages.

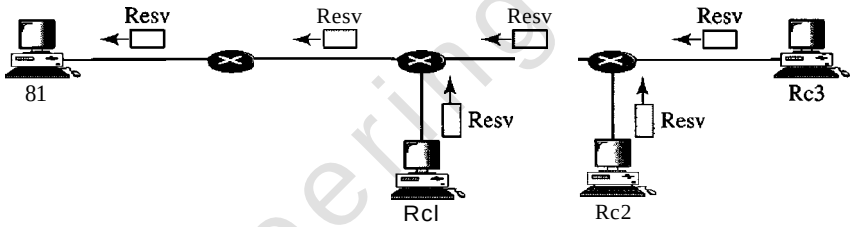
Resv Messages After a receiver has received a Path message, it sends a *Resv* message. The Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP. If a router does not support RSVP on the path, it routes

Figure 24.22 *Path messages*



the packet based on the best-effort delivery methods we discussed before. Figure 24.23 shows the Resv messages.

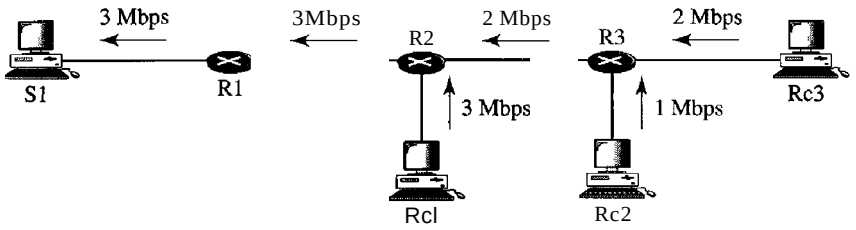
Figure 24.23 *Resv messages*



Reservation Merging

In RSVP, the resources are not reserved for each receiver in a flow; the reservation is merged. In Figure 24.24, Rc3 requests a 2-Mbps bandwidth while Rc2 requests a I-Mbps bandwidth. Router R3, which needs to make a bandwidth reservation, merges the two requests. The reservation is made for 2 Mbps, the larger of the two, because a 2-Mbps input reservation can handle both requests. The same situation is true for R2. The reader may ask why Rc2 and Rc3, both belonging to one single flow, request different amounts of bandwidth. The answer is that, in a multimedia environment, different receivers may handle different grades of quality. For example, Rc2 may be able to receive video only at 1 Mbps (lower quality), while Rc3 may be able to receive video at 2 Mbps (higher quality).

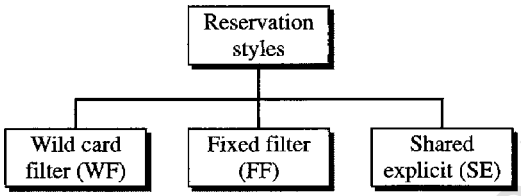
Figure 24.24 *Reservation merging*



Reservation Styles

When there is more than one flow, the router needs to make a reservation to accommodate all of them. RSVP defines three types of reservation styles, as shown in Figure 24.25.

Figure 24.25 *Reservation styles*



Wild Card Filter Style In this style, the router creates a single reservation for all senders. The reservation is based on the largest request. This type of style is used when the flows from different senders do not occur at the same time.

Fixed Filter Style In this style, the router creates a distinct reservation for each flow. This means that if there are *n* flows, *n* different reservations are made. This type of style is used when there is a high probability that flows from different senders will occur at the same time.

Shared Explicit Style In this style, the router creates a single reservation which can be shared by a set of flows.

Soft State

The reservation information (state) stored in every node for a flow needs to be refreshed periodically. This is referred to as a *soft state* as compared to the *hard state* used in other virtual-circuit protocols such as ATM or Frame Relay, where the information about the flow is maintained until it is erased. The default interval for refreshing is currently 30 s.

Problems with Integrated Services

There are at least two problems with Integrated Services that may prevent its full implementation in the Internet: scalability and service-type limitation.

Scalability

The Integrated Services model requires that each router keep information for each flow. As the Internet is growing every day, this is a serious problem.

Service-Type Limitation

The Integrated Services model provides only two types of services, guaranteed and control-load. Those opposing this model argue that applications may need more than these two types of services.

24.8 DIFFERENTIATED SERVICES

Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services. Two fundamental changes were made:

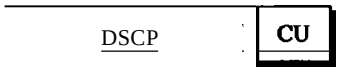
1. The main processing was moved from the core of the network to the edge of the network. This solves the scalability problem. The routers do not have to store information about flows. The applications, or hosts, define the type of service they need each time they send a packet.
2. The per-flow service is changed to per-class service. The router routes the packet based on the class of service defined in the packet, not the flow. This solves the service-type limitation problem. We can define different types of classes based on the needs of applications.

Differentiated Services is a class-based QoS model designed for IP.

DS Field

In Diffserv, each packet contains a field called the DS field. The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router. IETF proposes to replace the existing TOS (type of service) field in IPv4 or the class field in IPv6 by the DS field, as shown in Figure 24.26.

Figure 24.26 *DS field*



The DS field contains two subfields: DSCP and CU. The DSCP (Differentiated Services Code Point) is a 6-bit subfield that defines the per-hop behavior (PHB). The 2-bit CU (currently unused) subfield is not currently used.

The Diffserv capable node (router) uses the DSCP 6 bits as an index to a table defining the packet-handling mechanism for the current packet being processed.

Per-Hop Behavior

The Diffserv model defines per-hop behaviors (PHBs) for each node that receives a packet. So far three PHBs are defined: DE PHB, EF PHB, and AF PHB.

DE PHB The DE PHB (default PHB) is the same as best-effort delivery, which is compatible with TOS.

EF PHB The EF PHB (expedited forwarding PHB) provides the following services:

- Low loss

- Low latency
- Ensured bandwidth

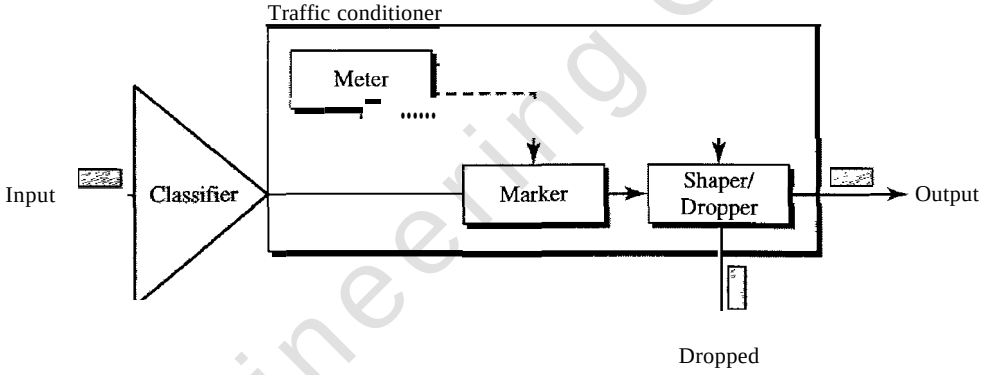
This is the same as having a virtual connection between the source and destination.

AF PHB The AF PHB (assured forwarding PHB) delivers the packet with a high assurance as long as the class traffic does not exceed the traffic profile of the node. The users of the network need to be aware that some packets may be discarded.

Traffic Conditioner

To implement Oiffserv, the OS node uses traffic conditioners such as meters, markers, shapers, and droppers, as shown in Figure 24.27.

Figure 24.27 *Traffic conditioner*



Meters The meter checks to see if the incoming flow matches the negotiated traffic profile. The meter also sends this result to other components. The meter can use several tools such as a token bucket to check the profile.

Marker A marker can remark a packet that is using best-effort delivery (OSCP: 000000) or down-mark a packet based on information received from the meter. Down-marking (lowering the class of the flow) occurs if the flow does not match the profile. A marker does not up-mark (promote the class) a packet.

Shaper A shaper uses the information received from the meter to reshape the traffic if it is not compliant with the negotiated profile.

Dropper A dropper, which works as a shaper with no buffer, discards packets if the flow severely violates the negotiated profile.

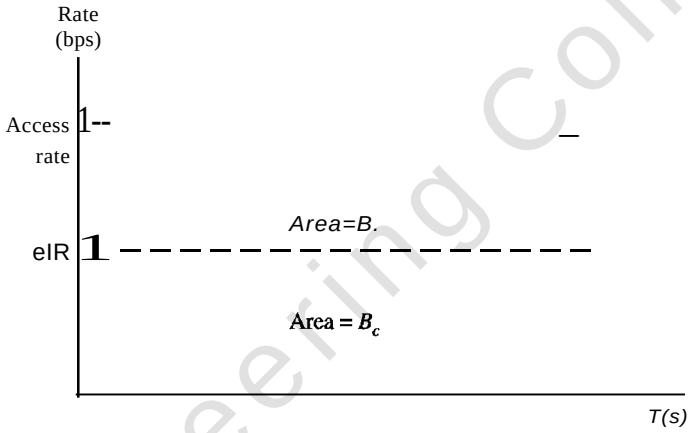
24.9 QoS IN SWITCHED NETWORKS

We discussed the proposed models for QoS in the IP protocols. Let us now discuss QoS as used in two switched networks: Frame Relay and ATM. These two networks are virtual-circuit networks that need a signaling protocol such as RSVP.

QoS in Frame Relay

Four different attributes to control traffic have been devised in Frame Relay: access rate, committed burst size *Be'* committed information rate (CIR), and excess burst size *Be'* These are set during the negotiation between the user and the network. For PVC connections, they are negotiated once; for SVC connections, they are negotiated for each connection during connection setup. Figure 24.28 shows the relationships between these four measurements.

Figure 24.28 *Relationship between traffic control attributes*



Access Rate

For every connection, an access rate (in bits per second) is defined. The access rate actually depends on the bandwidth of the channel connecting the user to the network. The user can never exceed this rate. For example, if the user is connected to a Frame Relay network by a T-l line, the access rate is 1.544 Mbps and can never be exceeded.

Committed Burst Size

For every connection, Frame Relay defines a committed burst size *Be.* This is the maximum number of bits in a predefined time that the network is committed to transfer without discarding any frame or setting the DE bit. For example, if a *Be* of 400 kbits for a period of 4 s is granted, the user can send up to 400 kbits during a 4-s interval without worrying about any frame loss. Note that this is not a rate defined for each second. It is a cumulative measurement. The user can send 300 kbits during the first second, no data during the second and the third seconds, and finally 100 kbits during the fourth second.

Committed Information Rate

The committed information rate (CIR) is similar in concept to committed burst size except that it defines an average rate in bits per second. If the user follows this rate continuously, the network is committed to deliver the frames. However, because it is an average measurement, a user may send data at a higher rate than the CIR at times or at

a lower rate other times. As long as the average for the predefined period is met, the frames will be delivered.

The cumulative number of bits sent during the predefined period cannot exceed B_c. Note that the CIR is not an independent measurement; it can be calculated by using the following formula:

$$\text{CIR} = \frac{B_c}{T} \text{ bps}$$

For example, if the *Be* is 5 kbits in a period of 5 s, the CIR is *5000/5,* or 1 kbps.

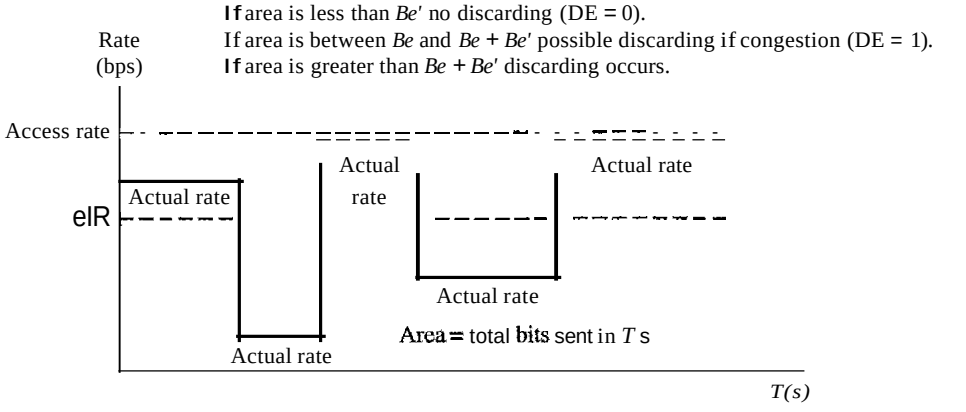
Excess Burst Size

For every connection, Frame Relay defines an excess burst size *Be"* This is the maximum number of bits in excess of *Be* that a user can send during a predefined time. The network is committed to transfer these bits if there is no congestion. Note that there is less commitment here than in the case of B_c. The network is committing itself conditionally.

User Rate

Figure 24.29 shows how a user can send bursty data. If the user never exceeds B_c, the network is committed to transmit the frames without discarding any. If the user exceeds *Be* by less than *Be* (that is, the total number of bits is less than *Be* + *Be)'* the network is committed to transfer all the frames if there is no congestion. If there is congestion, some frames will be discarded. The first switch that receives the frames from the user has a counter and sets the DE bit for the frames that exceed B_c. The rest of the switches will discard these frames if there is congestion. Note that a user who needs to send data faster may exceed the *Be* level. As long as the level is not above *Be* + *Be'* there is a chance that the frames will reach the destination without being discarded. Remember, however, that the moment the user exceeds the *Be* + *Be* level, all the frames sent after that are discarded by the first switch.

Figure 24.29 *User rate in relation to Be and Be* + *Be*



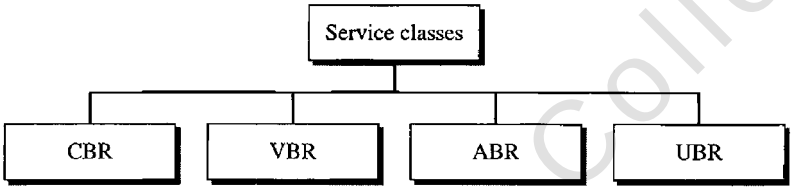
QoS in ATM

The QoS in ATM is based on the class, user-related attributes, and network-related attributes.

Classes

The ATM Forum defines four service classes: CBR, VBR, ABR, and UBR (see Figure 24.30).

Figure 24.30 *Service classes*



CBR The constant-bit-rate (CBR) class is designed for customers who need real-time audio or video services. The service is similar to that provided by a dedicated line such as a T line.

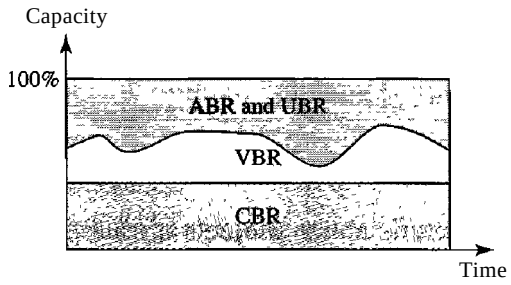
VBR The variable-bit-rate (VBR) class is divided into two subclasses: real-time (VBR-RT) and non-real-time (VBR-NRT). VBR-RT is designed for those users who need real-time services (such as voice and video transmission) and use compression techniques to create a variable bit rate. VBR-NRT is designed for those users who do not need real-time services but use compression techniques to create a variable bit rate.

ABR The available-bit-rate (ABR) class delivers cells at a minimum rate. If more network capacity is available, this minimum rate can be exceeded. ABR is particularly suitable for applications that are bursty.

UBR The unspecified-bit-rate (UBR) class is a best-effort delivery service that does not guarantee anything.

Figure 24.31 shows the relationship of different classes to the total capacity of the network.

Figure 24.31 *Relationship of service classes to the total capacity of the network*



User-Related Attributes

ATM defines two sets of attributes. User-related attributes are those attributes that define how fast the user wants to send data. These are negotiated at the time of contract between a user and a network. The following are some user-related attributes:

SCR The *sustained cell rate* (SCR) is the average cell rate over a long time interval. The actual cell rate may be lower or higher than this value, but the average should be equal to or less than the SCR.

PCR The *peak cell rate* (PCR) defines the sender's maximum cell rate. The user's cell rate can sometimes reach this peak, as long as the SCR is maintained.

MCR The *minimum cell rate* (MCR) defines the minimum cell rate acceptable to the sender. For example, if the MCR is 50,000, the network must guarantee that the sender can send at least 50,000 cells per second.

CVDT The *cell variation delay tolerance* (CVDT) is a measure of the variation in cell transmission times. For example, if the CVDT is 5 ns, this means that the difference between the minimum and the maximum delays in delivering the cells should not exceed 5 ns.

Network-Related Attributes

The network-related attributes are those that define characteristics of the network. The following are some network-related attributes:

CLR The *cell loss ratio* (CLR) defines the fraction of cells lost (or delivered so late that they are considered lost) during transmission. For example, if the sender sends 100 cells and one of them is lost, the CLR is

$$\text{CLR} = \frac{1}{100} = 10^{-2}$$

CTD The *cell transfer delay* (CTD) is the average time needed for a cell to travel from source to destination. The maximum CTD and the minimum CTD are also considered attributes.

CDV The *cell delay variation* (CDV) is the difference between the CTD maximum and the CTD minimum.

CER The *cell error ratio* (CER) defines the fraction of the cells delivered in error.

24.10 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

Congestion control and QoS are discussed in Sections 5.3 and 5.5 of [Tan03] and in Chapter 3 of [Sta9S]. Easy reading about QoS can be found in [FH98]. The full discussion of QoS is covered in [BlaOO].

24.11 KEY TERMS

access rate
additive increase
available bit rate (ABR)
average data rate
backward explicit congestion notification (BECN)
bursty data
choke packet
closed-loop congestion control
committed burst size *Be*
committed information rate (CIR)
congestion
congestion avoidance
congestion control
constant bit rate (CBR)
delay
Differentiated Services (DS or Diffserv)
effective bandwidth
excess burst size *Be*
first-in, first-out (FIFO) queuing
forward explicit congestion notification (FECN)
Integrated Services (IntServ)
jitter
leaky bucket
load
maximum burst size
multiplicative decrease
open-loop congestion control
Path message
peak data rate
per-hop behavior (PHB)
priority queuing
quality of service (QoS)
reliability
Resource Reservation Protocol (RSVP)
Resv message
slow start
throughput
token bucket
traffic shaping
unspecified bit rate (UBR)
variable bit rate (VBR)
weighted fair queuing

24.12 SUMMARY

- The average data rate, peak data rate, maximum burst size, and effective band width are qualitative values that describe a data flow.
- A data flow can have a constant bit rate, a variable bit rate, or traffic that is bursty.
- Congestion control refers to the mechanisms and techniques to control congestion and keep the load below capacity.
- Delay and throughput measure the performance of a network.
- Open-loop congestion control prevents congestion; closed-loop congestion control removes congestion.

- TCP avoids congestion through the use of two strategies: the combination of slow start and additive increase, and multiplicative decrease.
- Frame Relay avoids congestion through the use of two strategies: backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN).
- A flow can be characterized by its reliability, delay, jitter, and bandwidth.
- Scheduling, traffic shaping, resource reservation, and admission control are techniques to improve quality of service (QoS).
- FIFO queuing, priority queuing, and weighted fair queuing are scheduling techniques.
- Leaky bucket and token bucket are traffic shaping techniques.
- Integrated Services is a flow-based QoS model designed for IP.
- The Resource Reservation Protocol (RSVP) is a signaling protocol that helps IP create a flow and makes a resource reservation.
- Differential Services is a class-based QoS model designed for IP.
- Access rate, committed burst size, committed information rate, and excess burst size are attributes to control traffic in Frame Relay.
- Quality of service in ATM is based on service classes, user-related attributes, and network-related attributes.

24.13 PRACTICE SET

Review Questions

1. How are congestion control and quality of service related?
2. What is a traffic descriptor?
3. What is the relationship between the average data rate and the peak data rate?
4. What is the definition of bursty data?
5. What is the difference between open-loop congestion control and closed-loop congestion control?
6. Name the policies that can prevent congestion.
7. Name the mechanisms that can alleviate congestion.
8. What determines the sender window size in TCP?
9. How does Frame Relay control congestion?
10. What attributes can be used to describe a flow of data?
11. What are four general techniques to improve quality of service?
12. What is traffic shaping? Name two methods to shape traffic.
13. What is the major difference between Integrated Services and Differentiated Services?
14. How is Resource Reservation Protocol related to Integrated Services?
15. What attributes are used for traffic control in Frame Relay?
16. In regard to quality of service, how do user-related attributes differ from network-related attributes in ATM?

Exercises

17. The address field of a Frame Relay frame is 1011000000010111. Is there any congestion in the forward direction? Is there any congestion in the backward direction?
18. A frame goes from A to B. There is congestion in both directions. Is the PECN bit set? Is the BECN bit set?
19. In a leaky bucket used to control liquid flow, how many gallons of liquid are left in the bucket if the output rate is 5 gal/min, there is an input burst of 100 gal/min for 12 s, and there is no input for 48 s?
20. An output interface in a switch is designed using the leaky bucket algorithm to send 8000 bytes/s (tick). If the following frames are received in sequence, show the frames that are sent during each second.

    Frames 1, 2, 3, 4: 4000 bytes each

    Frames 5, 6, 7: 3200 bytes each

    Frames 8, 9: 400 bytes each

    Frames 10, 11, 12: 2000 bytes each
21. A user is connected to a Frame Relay network through a T-1 line. The granted CIR is 1 Mbps with a *Be* of 5 million bits/5 s and *Be* of 1 million bits/5 s.
    a. What is the access rate?
    b. Can the user send data at 1.6 Mbps?
    c. Can the user send data at 1 Mbps all the time? Is it guaranteed that frames are never discarded in this case?
    d. Can the user send data at 1.2 Mbps all the time? Is it guaranteed that frames are never discarded in this case? If the answer is no, is it guaranteed that frames are discarded only if there is congestion?
    e. Repeat the question in part (d) for a constant rate of 1.4 Mbps.
    f. What is the maximum data rate the user can use all the time without worrying about the frames being discarded?
    g. If the user wants to take a risk, what is the maximum data rate that can be used with no chance of discarding if there is no congestion?
22. In Exercise 21 the user sends data at 1.4 Mbps for 2 s and nothing for the next 3 s. Is there a danger of discarded data if there is no congestion? Is there a danger of discarded data if there is congestion?
23. In ATM, if each cell takes 10 μs to reach the destination, what is the CTD?
24. An ATM network has lost 5 cells out of 10,000 and 2 are in error. What is the CLR? What is the CER?

PART

6

Application Layer

Objectives

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, file access and transfer, access to system resources, surfing the world wide web, and network management.

The application layer is responsible for providing services to the user.

In this part, we briefly discuss some applications that are designed as a client/server pair in the Internet. The client sends a request for a service to the server; the server responds to the client.

Part 6 of the book is devoted to the application layer and the services provided by this layer.

Chapters

This part consists of five chapters: Chapters 25 to 29.

Chapter 25

Chapter 25 discusses the Domain Name System (DNS). DNS is a client/server application that provides name services for other applications. It enables the use of application-layer addresses, such as an email address, instead of network layer logical addresses.

Chapter 26

Chapter 26 discusses three common applications in the Internet: remote login, electronic mail, and file transfer. A remote login application allows the user to remotely access the resources of a system. An electronic mail application simulates the tasks of snail mail at a much higher speed. A file transfer application transfers files between remote systems.

Chapter 27

Chapter 27 discuss the ideas and issues in the famous world wide web (WWW). It also briefly describes the client/server application program (HTTP) that is commonly used to access the web.

Chapter 28

Chapter 28 is devoted to network management. We first discuss the general idea behind network management. We then introduce the client/server application, SNMP, that is used for this purpose in the Internet. Although network management can be implemented in every layer, the Internet has decided to use a client/server application.

Chapter 29

Chapter 29 discusses multimedia and a set of wiely-used application programs. These programs have generated new issues such as the need for new protocols in other layers to handle the specific problems related to multimedia. We briefly discuss these issues in this chapter.

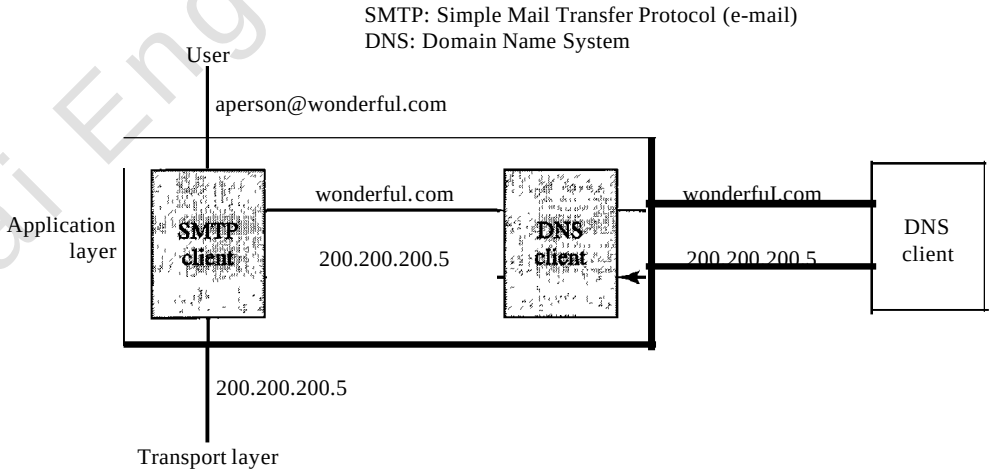
CHAPTER 25

Domain Name System

There are several applications in the application layer of the Internet model that follow the client/server paradigm. The client/server programs can be divided into two categories: those that can be directly used by the user, such as e-mail, and those that support other application programs. The Domain Name System (DNS) is a supporting program that is used by other programs such as e-mail.

Figure 25.1 shows an example of how a DNS client/server program can support an e-mail program to find the IP address of an e-mail recipient. A user of an e-mail program may know the e-mail address of the recipient; however, the IP protocol needs the IP address. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.

Figure 25.1 *Example of using the DNS service*



To identify an entity, TCPIIP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.

When the Internet was small, mapping was done by using a host file. The host file had only two columns: name and address. Every host could store the host file on its disk and update it periodically from a master host file. When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping.

Today, however, it is impossible to have one single host file to relate every address with a name and vice versa. The host file would be too large to store in every host. In addition, it would be impossible to update all the host files every time there was a change.

One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet.

Another solution, the one used today, is to divide this huge amount of information into smaller parts and store each part on a different computer. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS). In this chapter, we first discuss the concepts and ideas behind the DNS. We then describe the DNS protocol itself.

25.1 NAME SPACE

To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: fiat or hierarchical.

Flat Narne Space

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names mayor may not have a common section; if they do, it has no meaning. The main disadvantage of a fiat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical Narne Space

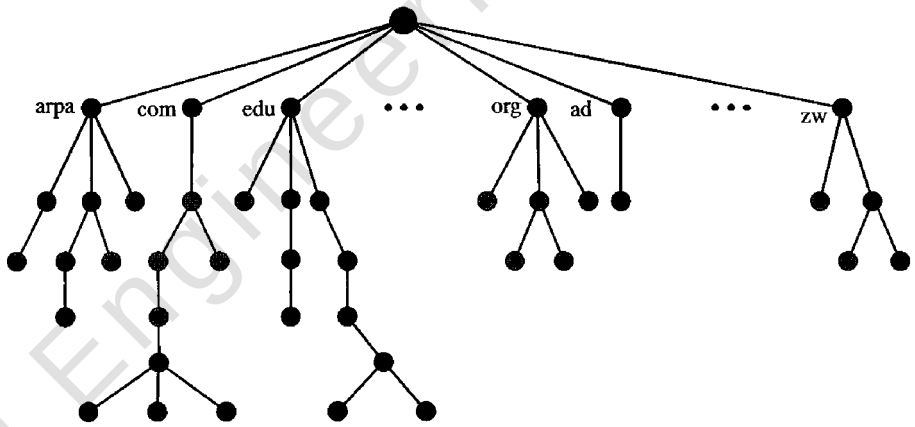
In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility of the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the

same, the whole address is different. For example, assume two colleges and a company call one of their computers *challenger.* The first college is given a name by the central authority such as *jhda.edu,* the second college is given the name *berkeley.edu,* and the company is given the name *smart.com.* When these organizations add the name *challenger* to the name they have already been given, the end result is three distinguishable names: *challenger.jhda.edu, challenger.berkeley.edu,* and *challenger.smart.com.* The names are unique without the need for assignment by a central authority. The central authority controls only part of the name, not the whole.

25.2 DOMAIN NAME SPACE

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127 (see Figure 25.2).

Figure 25.2 *Domain name space*



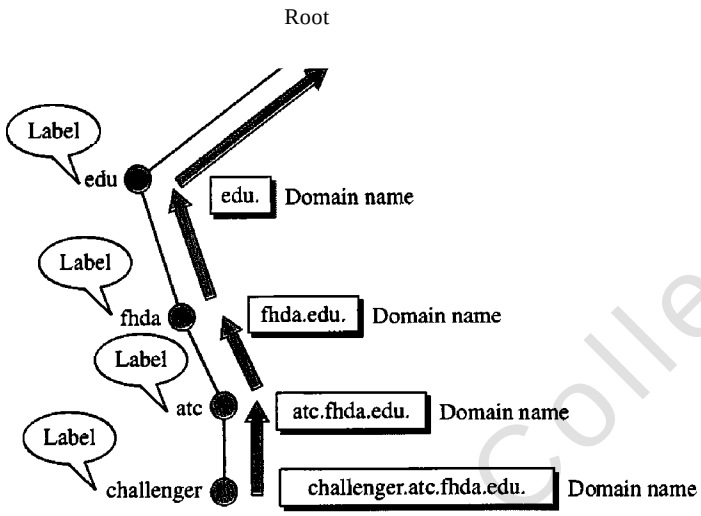
Label

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Figure 25.3 shows some domain names.

Figure 25.3 *Domain names and labels*



Fully Qualified Domain Name

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name

challenger.ate.tbda.edu.

is the FQDN of a computer named *challenger* installed at the Advanced Technology Center (ATC) at De Anza College. A DNS server can only match an FQDN to an address. Note that the name must end with a null label, but because null means nothing, the label ends with a dot (.).

Partially Qualified Domain Name

If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN. For example, if a user at the *jhda.edu.* site wants to get the IP address of the challenger computer, he or she can define the partial name

challenger

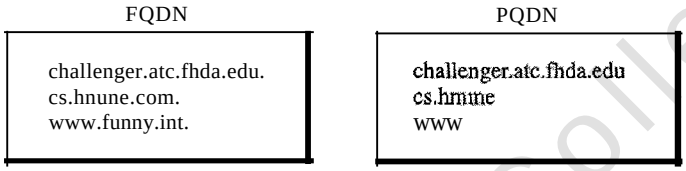
The DNS client adds the suffix *atc.jhda.edu.* before passing the address to the DNS server.

The DNS client normally holds a list of suffixes. The following can be the list of suffixes at De Anza College. The null suffix defines nothing. This suffix is added when the user defines an FQDN.

atc.fhda.edu
fhda.edu
null

Figure 25.4 shows some FQDNs and PQDNs.

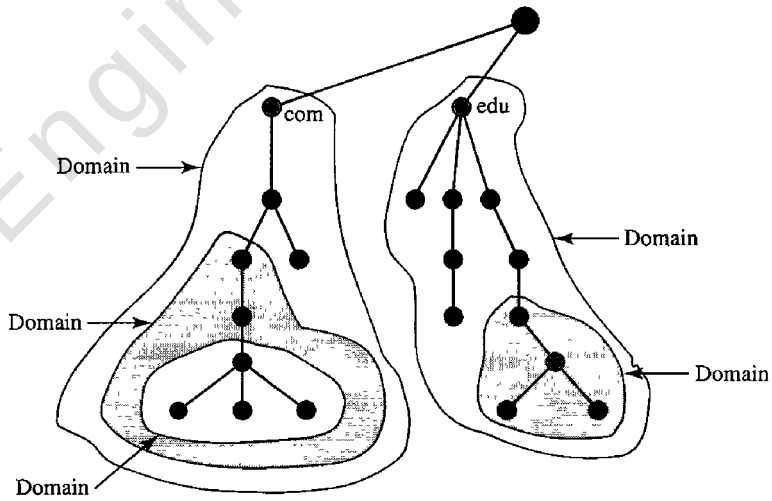
Figure 25.4 *FQDN and PQDN*



Domain

A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Figure 25.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).

Figure 25.5 *Domains*



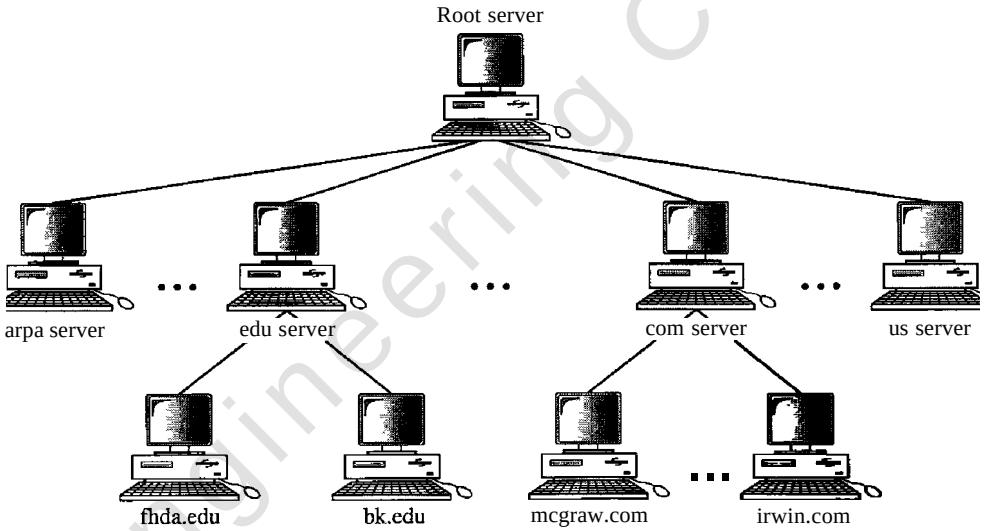
25.3 DISTRIBUTION OF NAME SPACE

The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not unreliable because any failure makes the data inaccessible.

Hierarchy of Name Servers

The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or a small domain. In other words, we have a hierarchy of servers in the same way that we have a hierarchy of names (see Figure 25.6).

Figure 25.6 *Hierarchy of name servers*

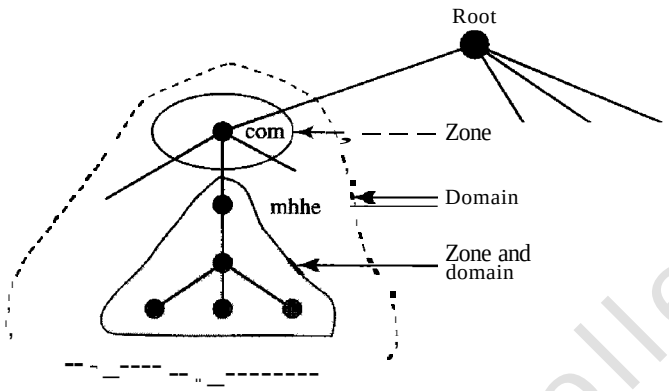


Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the *domain* and the *zone* refer to the same thing. The server makes a database called a *zone file* and keeps all the information for every node under that domain. However, if a server divides its domain into subdomains and delegates part of its authority to other servers, *domain* and *zone* refer to different things. The information about the nodes in the subdomains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers. Of course the original server does not free itself from responsibility totally: It still has a zone, but the detailed information is kept by the lower-level servers (see Figure 25.7).

A server can also divide part of its domain and delegate responsibility but still keep part of the domain for itself. In this case, its zone is made of detailed information for the part of the domain that is not delegated and references to those parts that are delegated.

Figure 25.7 *Zones and domains*



Root Server

A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space. The servers are distributed all around the world.

Primary and Secondary Servers

DNS defines two types of servers: primary and secondary. A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updaJing is required, it must be done by the primary server, which sends the updated version to the secondary.

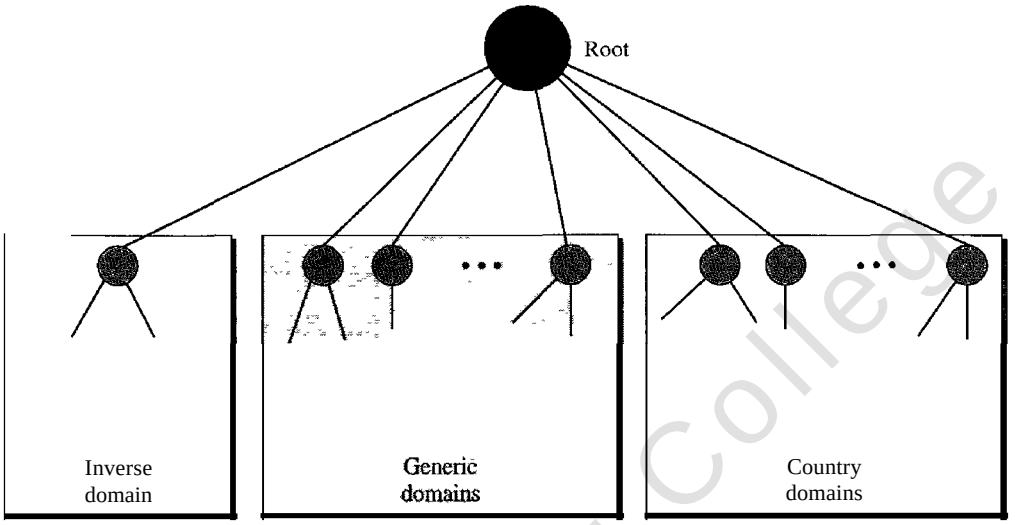
The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients. Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful to which zone we refer.

> A primary server loads all information from the disk file; the secondary server loads all information from the primary server. When the secondary downloads information from the primary, it is called zone transfer.

25.4 DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain (see Figure 25.8).

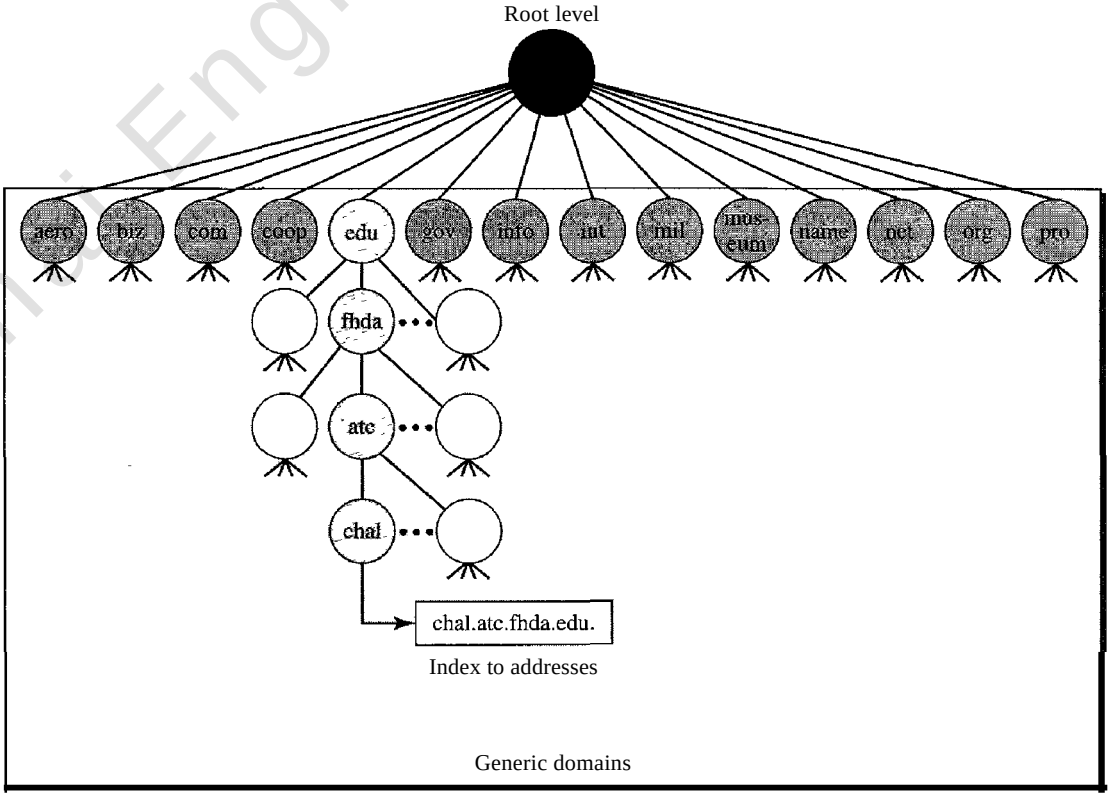
Figure 25.8 *DNS used in the Internet*



Generic Domains

The **generic domains** define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database (see Figure 25.9).

Figure 25.9 *Generic domains*



Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table 25.1.

Table 25.1 *Generic domain labels*

Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other nonprofit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).

Figure 25.10 shows the country domains section. The address *anza.cup.ca.us* can be translated to De Anza College in Cupertino, California, in the United States.

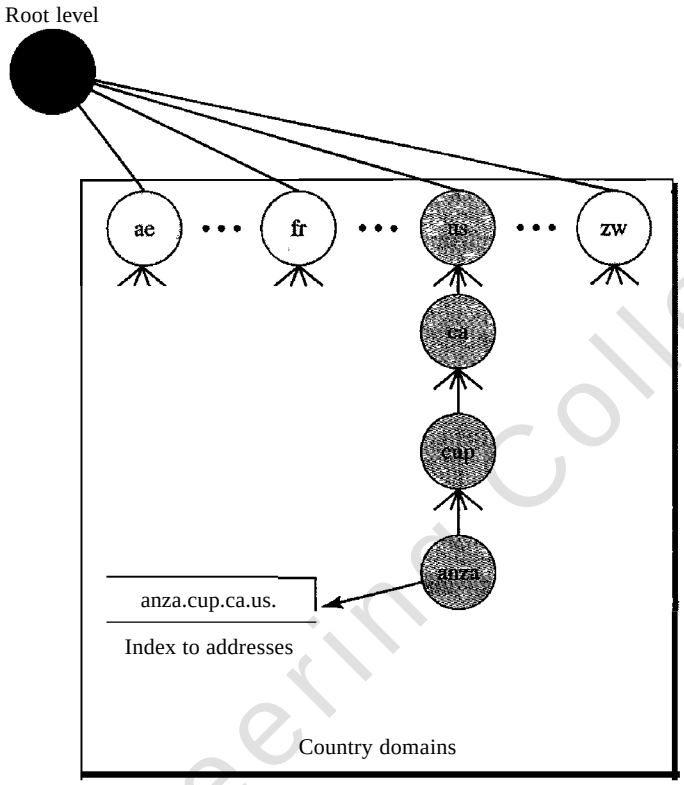
Inverse Domain

The inverse domain is used to map an address to a name. This may happen, for example, when a server has received a request from a client to do a task. Although the server has a file that contains a list of authorized clients, only the IP address of the client (extracted from the received IP packet) is listed. The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list.

This type of query is called an inverse or pointer (PTR) query. To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called *arpa* (for historical reasons). The second level is also one single node named *in-addr* (for inverse address). The rest of the domain defines IP addresses.

The servers that handle the inverse domain are also hierarchical. This means the netid part of the address should be at a higher level than the subnetid part, and the subnetid part

Figure 25.10 *Country domains*



higher than the hostid part. In this way, a server serving the whole site is at a higher level than the servers serving each subnet. This configuration makes the domain look inverted when compared to a generic or country domain. To follow the convention of reading the domain labels from the bottom to the top, an IF address such as 132.34.45.121 (a class B address with netid 132.34) is read as 121.45.34.132.in-addr. arpa. See Figure 25.11 for an illustration of the inverse domain configuration.

25.5 RESOLUTION

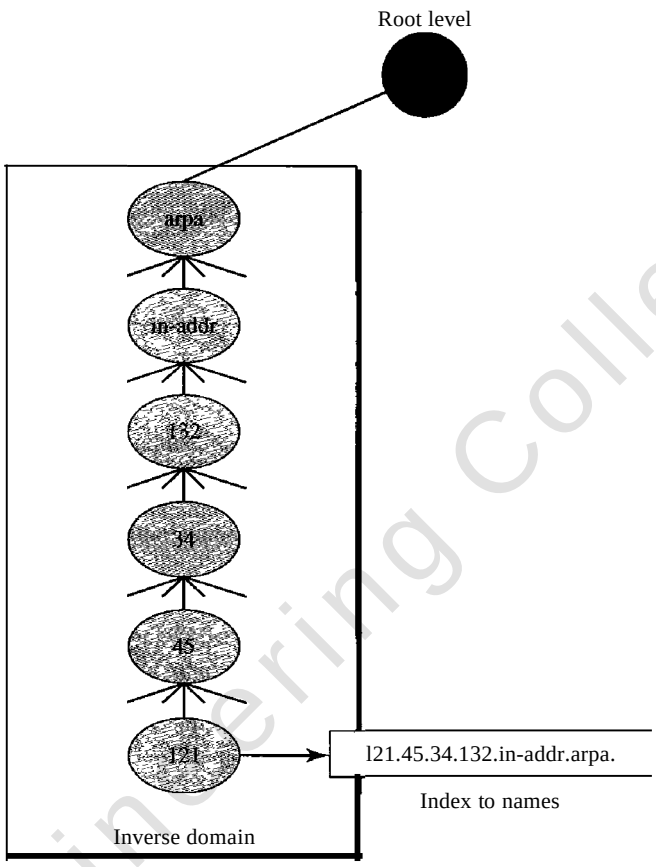
Mapping a name to an address or an address to a name is called *name-address resolution.*

Resolver

DNS is designed as a client/server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the infonnation.

After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

Figure 25.11 *Inverse domain*



Mapping Names to Addresses

Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the country domains to find the mapping.

If the domain name is from the generic domains section, the resolver receives a domain name such as *"chal.atc.jhda.edu."*. The query is sent by the resolver to the local DNS server for resolution. If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly.

If the domain name is from the country domains section, the resolver receives a domain name such as *"ch.jhda.cu.ca.us."*. The procedure is the same.

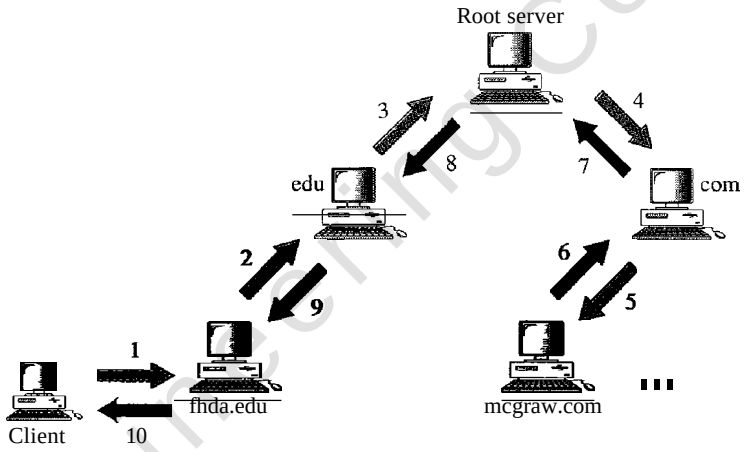
Mapping Addresses to Names

A client can send an IP address to a server to be mapped to a domain name. As mentioned before, this is called a PTR query. To answer queries of this kind, DNS uses the inverse domain. However, in the request, the IP address is reversed and the two labels *in-addr* and *arpa* are appended to create a domain acceptable by the inverse domain section. For example, if the resolver receives the IF address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is *"121.45.34.132.in-addr.arpa."* which is received by the local DNS and resolved.

Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution and is shown in Figure 25.12.

Figure 25.12 *Recursive resolution*



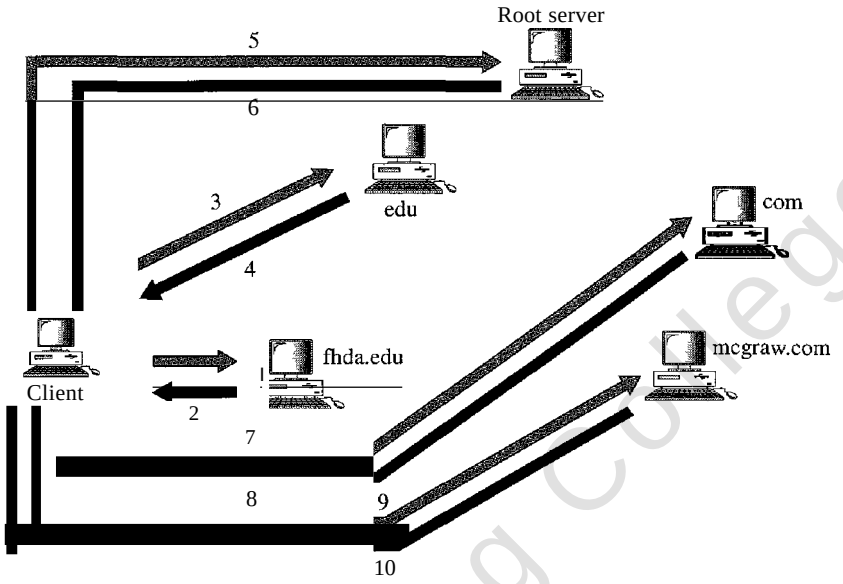
Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Now the client must repeat the query to the third server. This process is called iterative resolution because the client repeats the same query to multiple servers. In Figure 25.13 the client queries four servers before it gets an answer from the mcgraw.com server.

Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching. When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and solve the problem. However, to

Figure 25.13 *Iterative resolution*



inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as *unauthoritative.*

Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client. To counter this, two techniques are used. First, the authoritative server always adds information to the mapping called *time-to-live* (TTL). It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server. Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically, and those mappings with an expired TTL must be purged.

25.6 DNS MESSAGES

DNS has two types of messages: query and response. Both types have the same format. The query message consists of a header and question records; the response message consists of a header, question records, answer records, authoritative records, and additional records (see Figure 25.14).

Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is 12 bytes, and its format is shown in Figure 25.15.

The *identification* subfield is used by the client to match the response with the query. The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response. The *flags* subfield is a collection of

Figure 25.14 *Query and response messages*

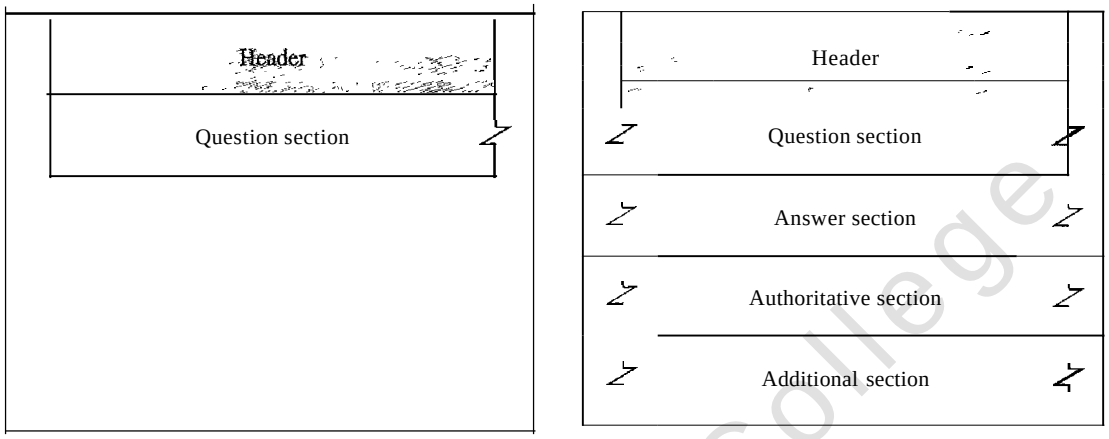


Figure 25.15 *Header format*

Identification	Flags
Number of question records	Number of answer records (all 0s in query message)
Number of authoritative records (all 0s in query message)	Number of additional records (all 0s in query message)

subfields that define the type of the message, the type of answer requested, the type of desired resolution (recursive or iterative), and so on. The *number of question records* subfield contains the number of queries in the question section of the message. The *number of answer records* subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message. The *number of authoritative records* subfield contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message. Finally, the *number of additional records* subfield contains the number additional records in the additional section of a response message. Its value is zero in the query message.

Question Section

This is a section consisting of one or more question records. It is present on both query and response messages. We will discuss the question records in a following section.

Answer Section

This is a section consisting of one or more resource records. It is present only on response messages. This section includes the answer from the server to the client (resolver). We will discuss resource records in a following section.

Authoritative Section

This is a section consisting of one or more resource records. It is present only on response messages. This section gives infonnation (domain name) about one or more authoritative servers for the query.

Additional Information Section

This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional infonnation that may help the resolver. For example, a server may give the domain name of an authoritative server to the resolver in the authoritative section, and include the IP address of the same authoritative server in the additional information section.

25.7 TYPES OF RECORDS

As we saw in Section 25.6, two types of records are used in DNS. The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

Question Record

A question record is used by the client to get information from a server. This contains the domain name.

- Resource Record

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records. Resource records are also what is returned by the server to the client.

25.8 REGISTRARS

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged.

Today, there are many registrars; their names and addresses can be found at

http://www.intenic.net

To register, the organization needs to give the name of its server and the IP address of the server. For example, a new commercial organization named *wonderful* with a server named *ws* and IP address 200.200.200.5 needs to give the following information to one of the registrars:

Domain name: ws.wonderful.com
IP address: 200.200.200.5

25.9 DYNAMIC DOMAIN NAME SYSTEM (DDNS)

When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.

The DNS master file must be updated dynamically. The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need. In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP (see Chapter 21) to a primary DNS server. The primary server updates the zone. The secondary servers are notified either actively or passively. In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes. In either case, after being notified about the change, the secondary requests information about the entire zone (zone transfer).

To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

25.10 ENCAPSULATION

DNS can use either UDP or TCP. In both cases the well-known port used by the server is port 53. UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit. If the size of the response message is more than 512 bytes, a TCP connection is used. In that case, one of two scenarios can occur:

- If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection. For example, if a secondary name server (acting as a client) needs a zone transfer from a primary server, it uses the TCP connection because the size of the information being transferred usually exceeds 512 bytes.
- If the resolver does not know the size of the response message, it can use the UDP port. However, if the size of the response message is more than 512 bytes, the server truncates the message and turns on the TC bit. The resolver now opens a TCP connection and repeats the request to get a full response from the server.

DNS can use the services of UDP or TCP using the well-known port 53.

25.11 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

DNS is discussed in [AL98], Chapter 17 of [For06], Section 9.1 of [PD03], and Section 7.1 of [Tan03].

Sites

The following sites are related to topics discussed in this chapter.

- www.intenic.net/Information about registrars
- www.ietf.org/rfc.html Information about RFCs

RFCs

The following RFCs are related to DNS:

799, 811, 819, 830, 881, 882, 883, 897, 920, 921, 1034,1035,1386,1480,1535,1536,1537, 1591, 1637, 1664, 1706, 1712, 1713, 1982, 2065, 2137, 2317, 2535, 2671

25.12 KEY TERMS

caching
country domain
DNS server
domain
domain name
domain name space
Domain Name System (DNS)
Dynamic Domain Name System (DDNS)
flat name space
fully qualified domain name (FQDN)
generic domain
hierarchical name space
host file
inverse domain
iterative resolution
label
name space
partially qualified domain name (PQDN)
primary server
query message
question record
recursive resolution
registrar
resolver
resource record
response message
root server
secondary server
subdomain
suffix
zone

25.13 SUMMARY

- The Domain Name System (DNS) is a client/server application that identifies each host on the Internet with a unique user-friendly name.
- DNS organizes the name space in a hierarchical structure to decentralize the responsibilities involved in naming.

- DNS can be pictured as an inverted hierarchical tree structure with one root node at the top and a maximum of 128 levels.
- Each node in the tree has a domain name.
- A domain is defined as any subtree of the domain name space.
- The name space information is distributed among DNS servers. Each server has jurisdiction over its zone.
- A root server's zone is the entire DNS tree.
- A primary server creates, maintains, and updates information about its zone.
- A secondary server gets its information from a primary server.
- The domain name space is divided into three sections: generic domains, country domains, and inverse domain.
- There are 14 generic domains, each specifying an organization type.
- Each country domain specifies a country.
- The inverse domain finds a domain name for a given IP address. This is called address-to-name resolution.
- Name servers, computers that run the DNS server program, are organized in a hierarchy.
- The DNS client, called a resolver, maps a name to an address or an address to a name.
- In recursive resolution, the client sends its request to a server that eventually returns a response.
- In iterative resolution, the client may send its request to multiple servers before getting an answer.
- Caching is a method whereby an answer to a query is stored in memory (for a limited time) for easy access to future requests.
- A fully qualified domain name (FQDN) is a domain name consisting of labels beginning with the host and going back through each level to the root node.
- A partially qualified domain name (PQDN) is a domain name that does not include all the levels between the host and the root node.
- There are two types of DNS messages: queries and responses.
- There are two types of DNS records: question records and resource records.
- Dynamic DNS (DDNS) automatically updates the DNS master file.
- DNS uses the services of UDP for messages of less than 512 bytes; otherwise, TCP is used.

25.14 PRACTICE SET

Review Questions

1. What is an advantage of a hierarchical name space over a flat name space for a system the size of the Internet?
2. What is the difference between a primary server and a secondary server?
3. What are the three domains of the domain name space?

4. What is the purpose of the inverse domain?
5. How does recursive resolution differ from iterative resolution?
6. What is an FQDN?
7. What is a PQDN?
8. What is a zone?
9. How does caching increase the efficiency of name resolution?
10. What are the two main categories ofDNS messages?
11. Why was there a need for DDNS?

Exercises

12. Determine which of the following is an FQDN and which is a PQDN.
    a. xxx
    b. xxx.yyy.
    c. xxx.yyy.net
    d. zzz.yyy.xxx.edu.
13. Determine which of the following is an FQDN and which is a PQDN.
    a. mil.
    b. edu.
    c. xxx.yyy.net
    d. zzz.yyy.xxx.edu
14. Which domain is used by your system, generic or country?
15. Why do we need a DNS system when we can directly use an IP address?
16. To find the IP address of a destination, we need the service of DNS. DNS needs the service of UDP or TCP. UDP or TCP needs the service of IP. IP needs an IP destination address. Is this a vicious cycle here?
17. If a DNS domain name is *voyager.fhda.edu,* how many labels are involved here? How many levels of hierarchy?
18. Is a PQDN necessarily shorter than the corresponding FQDN?
19. A domain name is *hello.customer.info.* Is this a generic domain or a country domain?
20. Do you think a recursive resolution is normally faster than an interactive one? Explain.
21. Can a query message have one question section but the corresponding response message have several answer sections?

CHAPTER 26

Remote Logging, Electronic Mail, and File Transfer

The main task of the Internet is to provide services for users. Among the most popular applications are remote logging, electronic mail, and file transfer. We discuss these three applications in this chapter; we discuss another popular use of the Internet, accessing the World Wide Web, in Chapter 27.

26.1 REMOTE LOGGING

In the Internet, users may want to run application programs at a remote site and create results that can be transferred to their local site. For example, students may want to connect to their university computer lab from their home to access application programs for doing homework assignments or projects. One way to satisfy that demand and others is to create a client/server application program for each desired service. Programs such as file transfer programs (FTPs), e-mail (SMTP), and so on are currently available. However, it would be impossible to write a specific client/server program for each demand.

The better solution is a general-purpose client/server program that lets a user access any application program on a remote computer; in other words, allow the user to log on to a remote computer. After logging on, a user can use the services available on the remote computer and transfer the results back to the local computer.

TELNET

In this section, we discuss such a client/server application program: TELNET. TELNET is an abbreviation for *TErminaL NETwork.* It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO). TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

TELNET is a general-purpose client/server application program.

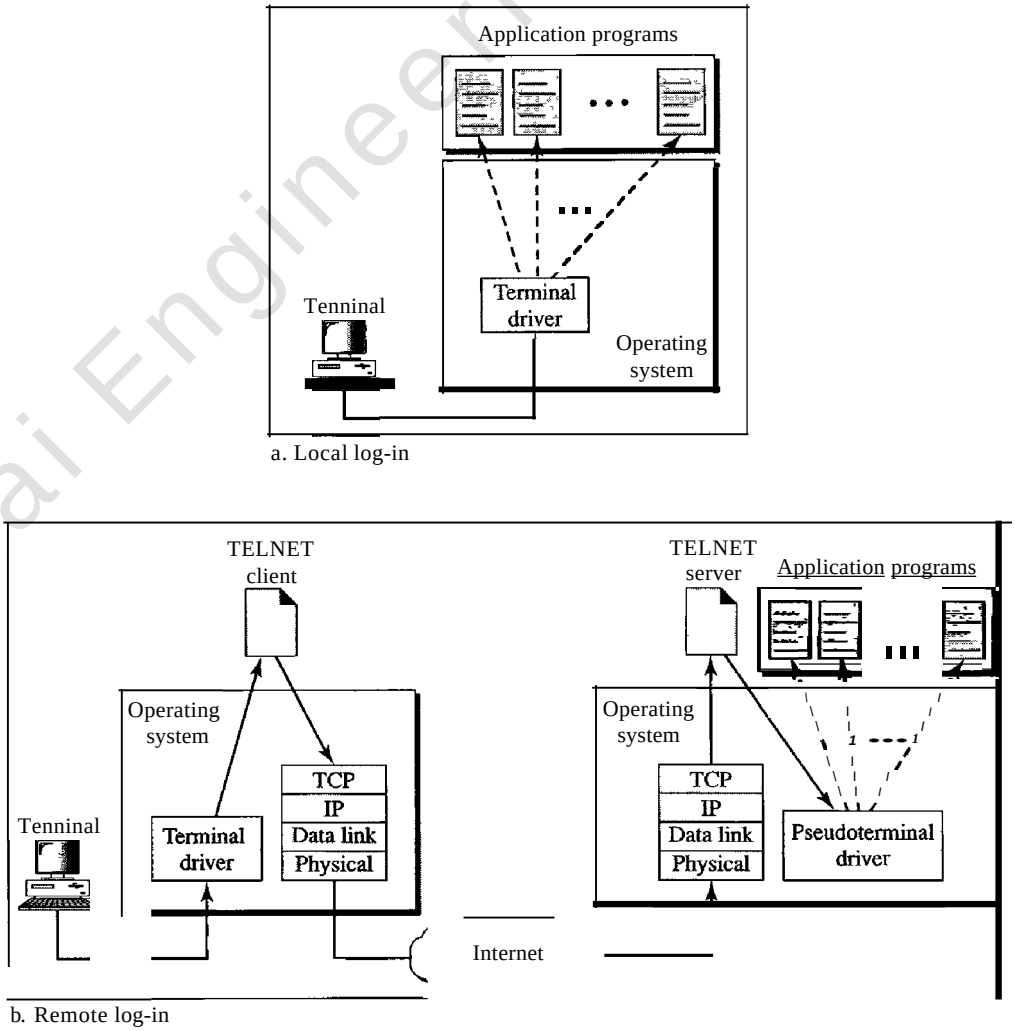
Timesharing Environment

TELNET was designed at a time when most operating systems, such as UNIX, were operating in a timesharing environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse. Even a microcomputer can simulate a terminal with a terminal emulator.

Logging

In a timesharing environment, users are part of the system with some right to access resources. Each authorized user has an identification and probablY,a password. The user identification defines the user as part of the system. To access the system, the user logs into the system with a user id or log-in name. The system also includes password checking to prevent an unauthorized user from accessing the resources. Figure 26.1 shows the logging process.

Figure 26.1 *Local and remote log-in*



When a user logs into a local timesharing system, it is called local log-in. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

When a user wants to access an application program or utility located on a remote machine, she performs remote log-in. Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters to a universal character set called *network virtual terminal (NVT) characters* and delivers them to the local *TCP/IP* protocol stack.

The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver. The solution is to add a piece of software called a *pseudoterminal driver* which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

Network Virtual Terminal

The mechanism to access a remote computer is complex. This is so because every computer and its operating system accept a special combination of characters as tokens. For example, the end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d.

We are dealing with heterogeneous systems. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the network virtual terminal (NVT) character set. Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer. For an illustration of this concept, see Figure 26.2.

NVT Character Set NVT uses two sets of characters, one for data and the other for control. Both are 8-bit bytes. For data, NVT is an 8-bit character set in which the 7 lowest-order bits are the same as ASCII and the highest-order bit is 0. To send control characters between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest-order bit is set to 1.

Table 26.1 lists some of the control characters and their meanings.

Figure 26.2 *Concept of NVT*

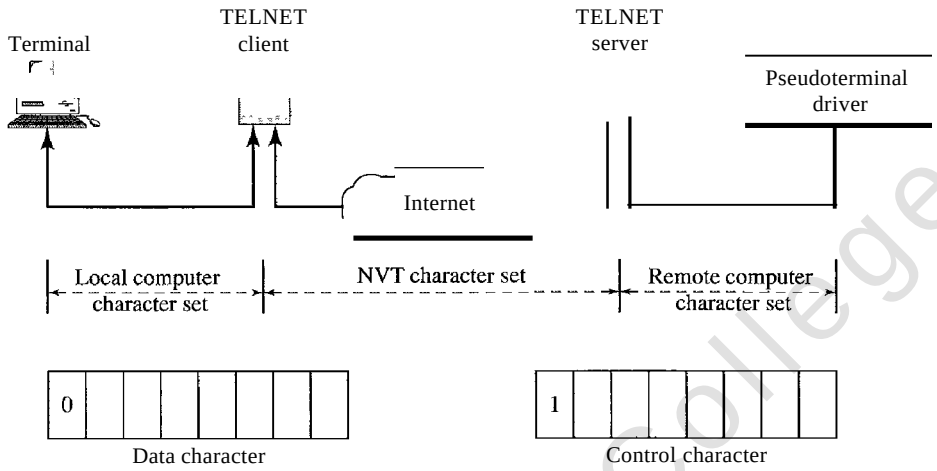


Table 26.1 *Some NVT control characters*

Character	*Decimal*	*Binary*	*Meaning*
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
lAC	255	11111111	Interpret (the next character) as control

Embedding

TELNET uses only one TCP connection. The server uses the well-known port 23, and the client uses an ephemeral port. The same connection is used for sending both data and

control characters. TELNET accomplishes this by embedding the control characters in the data stream. However, to distinguish data from control characters, each sequence of control characters is preceded by a special control character called *interpret as control* (lAC). For example, imagine a user wants a server to display a file *(filel)* on a remote server. She can type

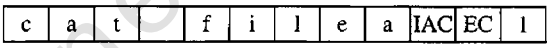
catfilel

However, suppose the name of the file has been mistyped *(filea* instead of *filel)*. The user uses the backspace key to correct this situation.

catjllea<backspace>l

However, in the default implementation of TELNET, the user cannot edit locally; the editing is done at the remote server. The backspace character is translated into two remote characters (lAC EC), which are embedded in the data and sent to the remote server. What is sent to the server is shown in Figure 26.3.

Figure 26.3 *An example ofembedding*



Typed at the remote terminal

Options

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal. Users with simpler terminals can use default features. Some control characters discussed previously are used to define options. Table 26.2 shows some common options.

Table 26.2 *Options*

Code	*Option*	*Meaning*
0	Binary	Interpret as 8-bit binary transmission.
1	Echo	Echo the data received on one side to the other.
3	Suppress go ahead	Suppress go-ahead signals after data.
5	Status	Request the status of TELNET.
6	Timing mark	Define the timing marks.
24	Terminal type	Set the terminal type.
32	Terrninalspeed	Set the terminal speed.
34	Line mode	Change to line mode.

Option Negotiation To use any of the options mentioned in the previous section first requires option negotiation between the client and the server. Four control characters are used for this purpose; these are shown in Table 26.3.

Table 26.3 *NVT character set for option negotiation*

Character	*Decimal*	*Binary*	*Meaning*
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

A party can offer to enable or disable an option if it has the right to do so. The offering can be approved or disapproved by the other party. To offer enabling, the offering party sends the WILL command, which means "Will I enable the option?" The other party sends either the DO command, which means "Please do," or the DONT command, which means "Please don't." To offer disabling, the offering party sends the WONT command, which means "I won't use this option any more." The answer must be the DONT command, which means "Don't use it anymore."

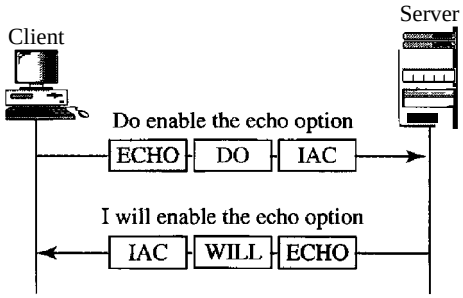
A party can request from the other party the enabling or the disabling of an option. To request enabling, the requesting party sends the DO command, which means "Please do enable the option." The other party sends either the WILL command, which means "I will," or the WONT command, which means "I won't." To request disabling, the requesting party sends the DONT command, which means "Please don't use this option anymore." The answer must be the WONT command, which means "I won't use it anymore."

Example 26.1

Figure 26.4 shows an example of option negotiation. In this example, the client wants the server to echo each character sent to the server. In other words, when a character is typed at the user keyboard terminal, it goes to the server and is sent back to the screen of the user before being processed. The echo option is enabled by the server because it is the server that sends the characters back to the user tenninal. Therefore, the client should *request* from the server the enabling of the option using DO. The request consists of three characters: lAC, DO, and ECHO. The server accepts the request and enables the option. It informs the client by sending the three-character approval: lAC, WILL, and ECHO.

Suboption Negotiation Some options require additional information. For example, to define the type or speed of a terminal, the negotiation includes a string or a number

Figure 26.4 *Example* 26.1: *Echo option*



to define the type or speed. In either case, the two suboption characters indicated in Table 26.4 are needed for suboption negotiation.

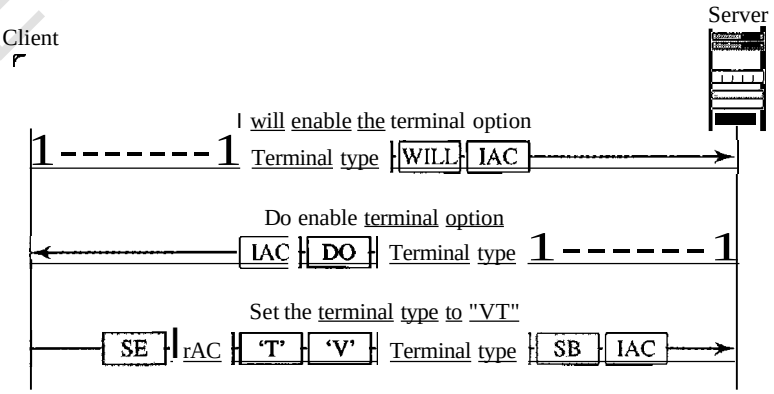
Table 26.4 *NVT character set for suboption negotiation*

Character	*Decimal*	*Binary*	*Meaning*
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

Example 26.2

Figure 26.5 shows an example of suboption negotiation. In this example, the client wants to negotiate the type of the terminal.

Figure 26.5 *Example of suboption negotiation*



Mode of Operation

Most TELNET implementations operate in one of three modes: default mode, character mode, or line mode.

Default Mode The default mode is used if no other modes are invoked through option negotiation. In this mode, the echoing is done by the client. The user types a

character, and the client echoes the character on the screen (or printer) but does not send it until a whole line is completed.

Character Mode In the character mode, each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed if the transmission time is long (such as in a satellite connection). It also creates overhead (traffic) for the network because three TCP segments must be sent for each character of data.

Line Mode A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode. In this mode, called the line mode, line editing (echoing, character erasing, line erasing, and so on) is done by the client. The client then sends the whole line to the server.

26.2 ELECTRONIC MAIL

One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program. Its architecture consists of several components that we discuss in this chapter.

At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

In this chapter, we first study the general architecture of an e-mail system including the three main components: user agent, message transfer agent, and message access agent. We then describe the protocols that implement these components.

Architecture

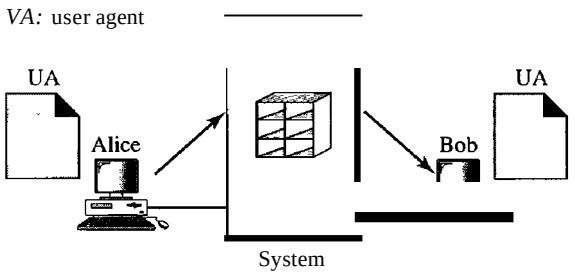
To explain the architecture of e-mail, we give four scenarios. We begin with the simplest situation and add complexity as we proceed. The fourth scenario is the most common in the exchange of email.

First Scenario

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same system; they are directly connected to a shared system. The administrator has created one mailbox for each user where the received messages are stored. A *mailbox* is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, Alice runs a *user agent (VA)* program to prepare the message and store it in Bob's mailbox. The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience, using a user agent. Figure 26.6 shows the concept.

This is similar to the traditional memo exchange between employees in an office. There is a mailroom where each employee has a mailbox with his or her name on it.

Figure 26.6 *First scenario in electronic mail*



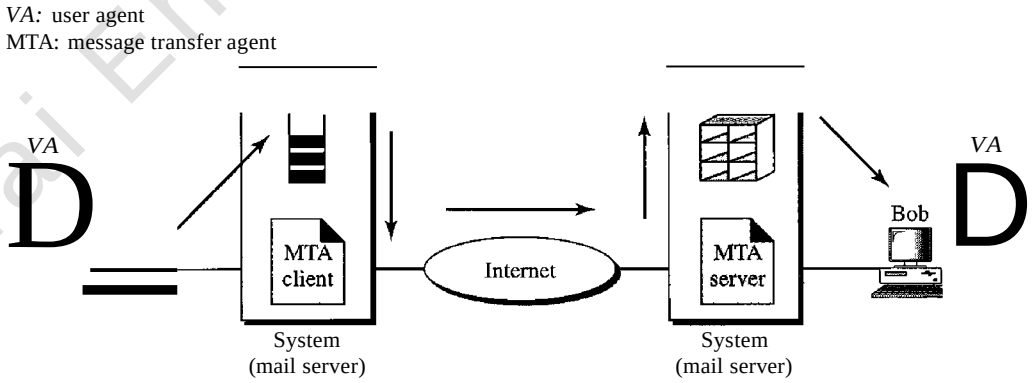
When Alice needs to send a memo to Bob, she writes the memo and inserts it into Bob's mailbox. When Bob checks his mailbox, he finds Alice's memo and reads it.

When the sender and the receiver of an e-mail are on the same system, we need only two user agents.

Second Scenario

In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systems. The message needs to be sent over the Internet. Here we need user agents (VAs) and message transfer agents (MTAs), as shown in Figure 26.7.

Figure 26.7 *Second scenario in electronic mail*



Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her site uses a queue to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one client and one server. Like most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a

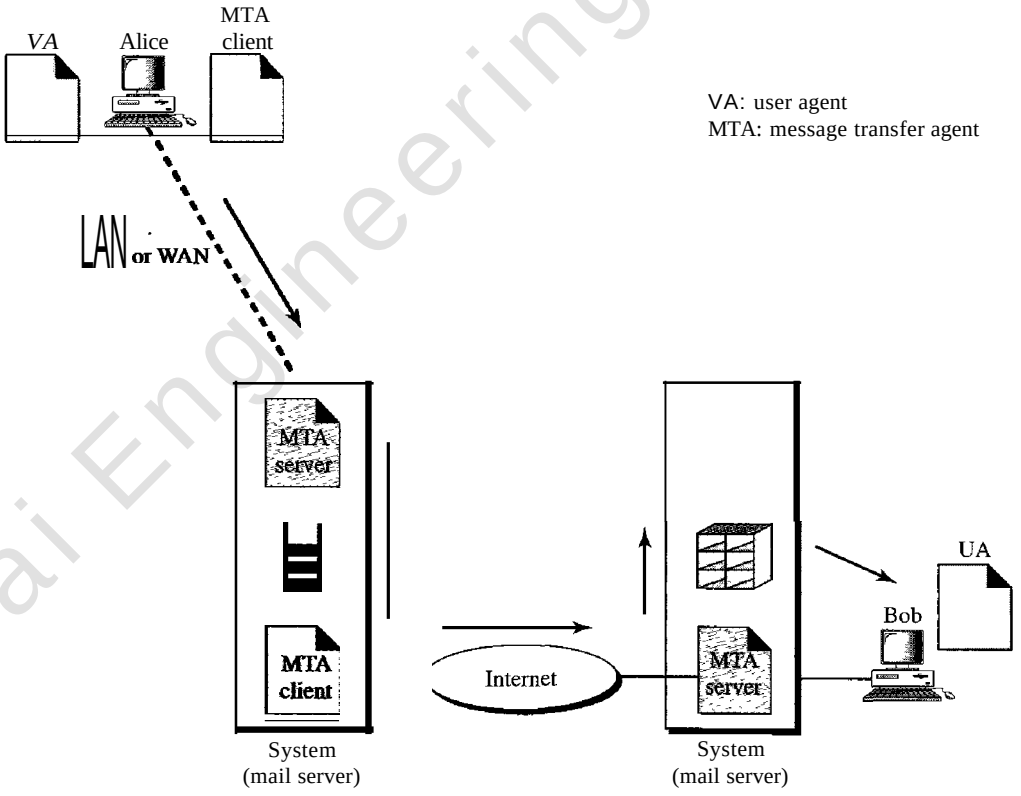
connection. The client, on the other hand, can be alerted by the system when there is a message in the queue to be sent.

When the sender and the receiver of an e-mail are on different systems, we need two VAs and a pair of MTAs (client and server).

Third Scenario

In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for handling e-mails-all users need to send their messages to this mail server. Figure 26.8 shows the situation.

Figure 26.8 *Third scenario in electronic mail*



Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in tum, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox.

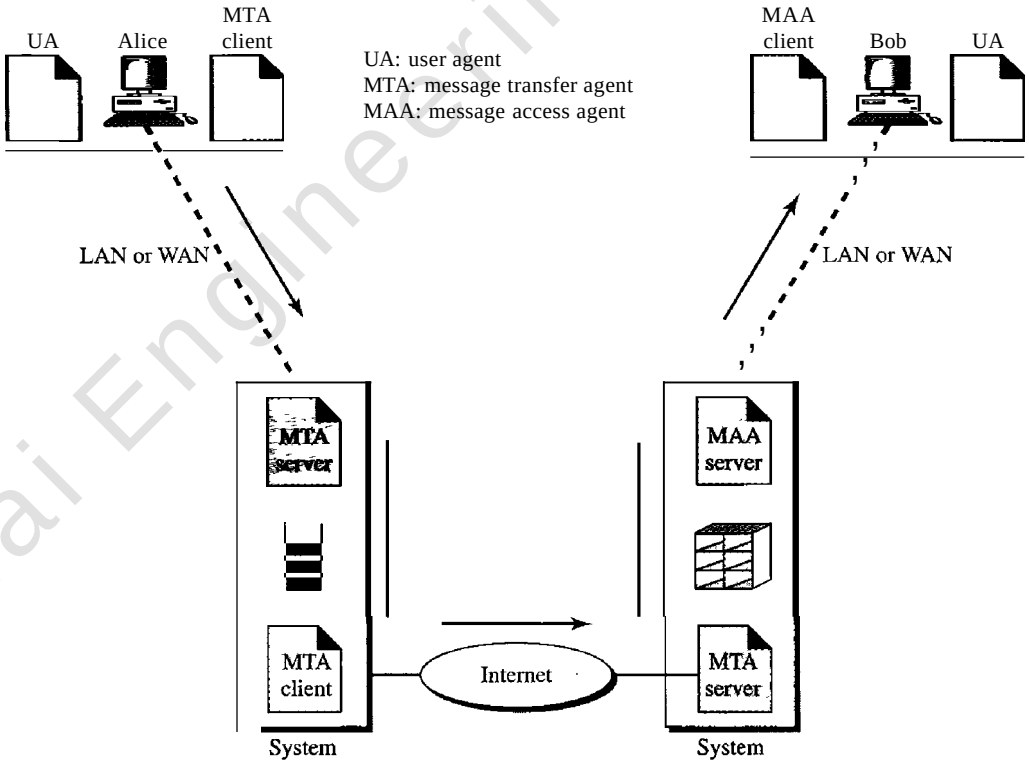
At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client/server programs.

When the sender is connected to the mail server via a LAN or a WAN, we need two *VAs* and two paIrs of MTAs (clIent and server).

Fourth Scenario

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it. Here, we need another set of client/server agents, which we call message access agents (MAAs). Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages. The situation is shown in Figure 26.9.

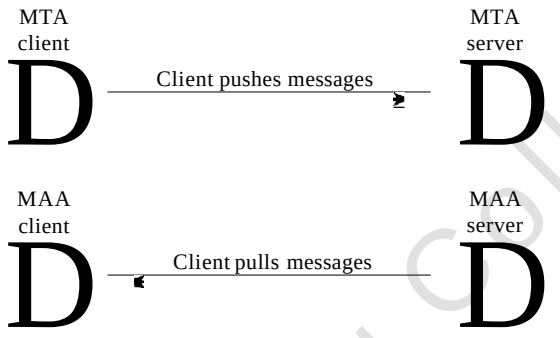
Figure 26.9 *Fourth scenario in electronic mail*



There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a-WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

Second, note that Bob needs another pair of client/server programs: message access programs. This is so because an MTA client/server program is a *push* program: the client pushes the message to the server. Bob needs a *pull* program. The client needs to pull the message from the server. Figure 26.10 shows the difference.

Figure 26.10 *Push versus pull in electronic email*



When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two VAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server). This is the most common situation today.

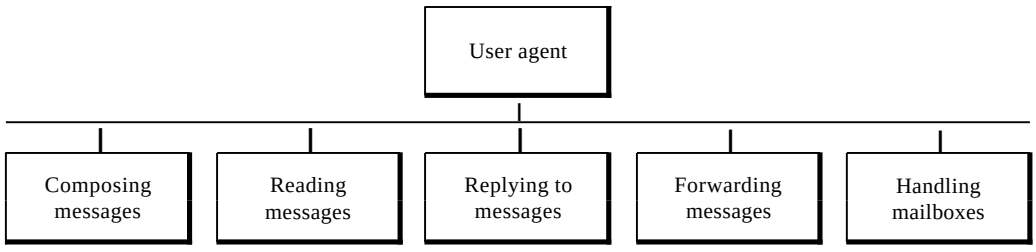
User Agent

The first component of an electronic mail system is the user agent *(VA).* It provides service to the user to make the process of sending and receiving a message easier.

Services Provided by a User Agent

A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles mailboxes. Figure 26.11 shows the services of a typical user agent.

Figure 26.11 *Services of user agent*



Composing Messages A user agent helps the user compose the e-mail message to be sent out. Most user agents provide a template on the screen to be filled in by the user. Some even have a built-in editor that can do spell checking, grammar checking, and

other tasks expected from a sophisticated word processor. A user, of course, could alternatively use his or her favorite text editor or word processor to create the message and import it, or cut and paste it, into the user agent template.

Reading Messages The second duty of the user agent is to read the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Most user agents show a one-line summary of each received mail. Each e-mail contains the following fields.

1. A number field.
2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to.
3. The size of the message.
4. The sender.
5. The optional subject field.

Replying to Messages After reading a message, a user can use the user agent to reply to a message. A user agent usually allows the user to reply to the original sender or to reply to all recipients of the message. The reply message may contain the original message (for quick reference) and the new message.

Forwarding Messages *Replying* is defined as sending a message to the sender or recipients of the copy. *Forwarding* is defined as sending the message to a third party. A user agent allows the receiver to forward the message, with or without extra comments, to a third party.

Handling Mailboxes

A user agent normally creates two mailboxes: an inbox and an outbox. Each box is a file with a special format that can be handled by the user agent. The inbox keeps all the received e-mails until they are deleted by the user. The outbox keeps all the sent e-mails until the user deletes them. Most user agents today are capable of creating customized mailboxes.

User Agent Types

There are two types of user agents: command-driven and GUI-based.

Command-Driven Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients. Some examples of command-driven user agents are *mail, pine,* and *elm.*

Some examples of command-driven user agents are *mail, pine,* and *elm.*

GUI-Based Modem user agents are GUI-based. They contain graphical-user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu

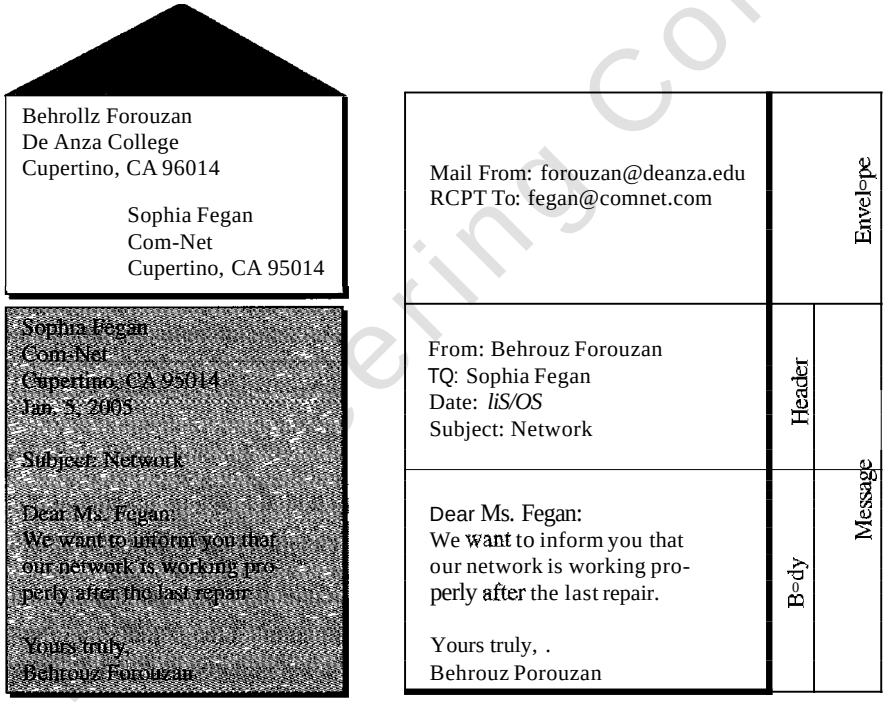
bars, and windows that make the services easy to access. Some examples of GUI-based user agents are Eudora, Microsoft's Outlook, and Netscape.

Some examples of GUI·based user agents are *Eudora, Outlook,* and *Netscape.*

Sending Mail

To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an *envelope* and a *message* (see Figure 26.12).

Figure 26.12 *Format of an e-mail*



a. Postal mail b. Electromc mall

Envelope The envelope usually contains the sender and the receiver addresses.

Message The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information (such as encoding type, as we see shortly). The body of the message contains the actual information to be read by the recipient.

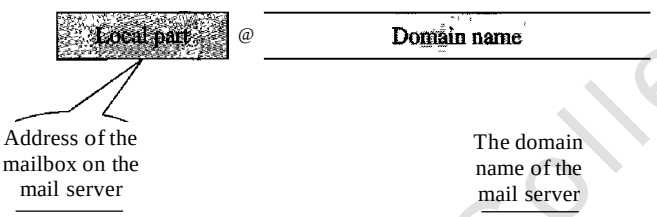
Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the *VA* informs the user with a notice. If the user is ready to read themail.alist is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a local part and a domain name, separated by an @ sign (see Figure 26.13).

Figure 26.13 *E-mail address*



Local Part The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.

Domain Name The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; the hosts are sometimes called *mail servers* or *exchangers.* The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (for example, the name of the organization).

Mailing List

Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA. If there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity.

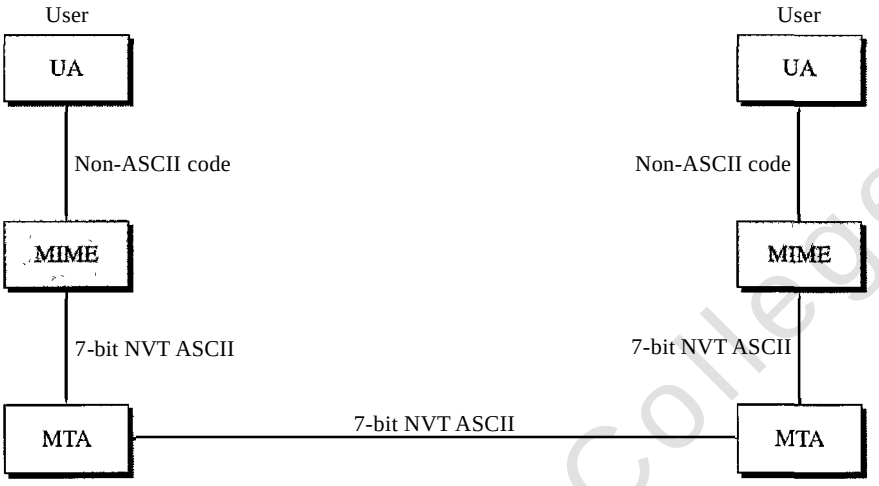
MIME

Electronic mail has a simple structure. Its simplicity, however, comes at a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data.

We can think of MIME as a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa, as shown in Figure 26.14.

Figure 26.14 *MIME*

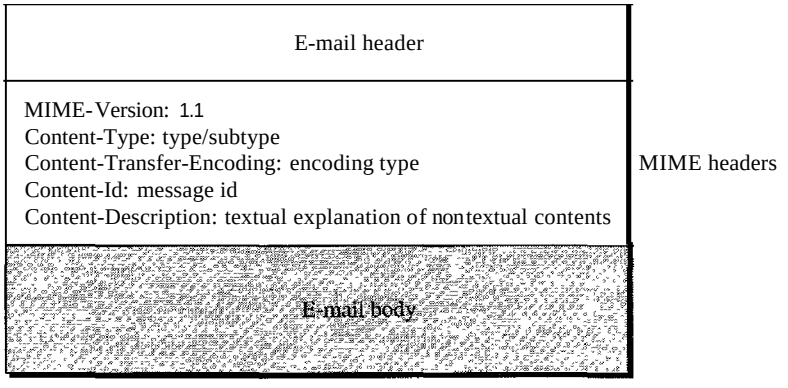


MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:

1. MIME-Version
2. Content-Type
3. Content-Transfer-Encoding
4. Content-Id
5. Content-Description

Figure 26.15 shows the MIME headers. We will describe each header in detail.

Figure 26.15 *MIME header*



MIME-Version This header defines the version of MIME used. The current version is 1.1.

MIME-Version: 1.1

Content-Type This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters.

Content-Type: <type */* subtype; parameters>

MIME allows seven different types of data. These are listed in Table 26.5.

Table 26.5 *Data types and subtypes in MIME*

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Chapter 27)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to mixed subtypes, but the default is message/ RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	IPEG	Image is in IPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

Content-Transfer-Encoding This header defines the method used to encode the messages into 0s and Is for transport:

Content-Transfer-Encoding: <type>

The five types of encoding methods are listed in Table 26.6.

Content-Id This header uniquely identifies the whole message in a multiple-message environment.

Content-Id: id=<content-id>

Table 26.6 *Content-transfer-encoding*

Type	*Description*
7-bit	NVT ASCII characters and short lines
8-bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base-64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equals sign followed by an ASCII code

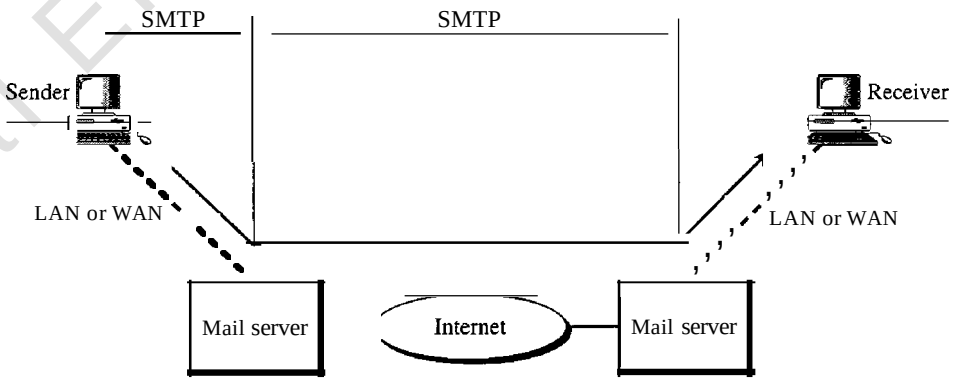
Content-Description This header defines whether the body is image, audio, or video.

Content-Description: <description>

Message Transfer Agent: SMTP

The actual mail transfer is done through message transfer agents. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP). As we said before, two pairs of MTA client/server programs are used in the most common situation (fourth scenario). Figure 26.16 shows the range of the SMTP protocol in this scenario.

Figure 26.16 *SMTP range*



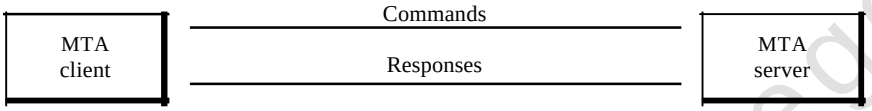
SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server (see Figure 26.17).

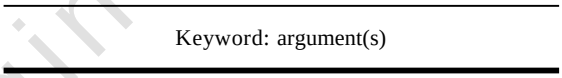
Figure 26.17 *Commands and responses*



Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Commands Commands are sent from the client to the server. The fonnat of a command is shown in Figure 26.18. It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands. The first five are mandatory; every implementation must support these five commands. The next three are often used and highly recommended. The last six are seldom used.

Figure 26.18 *Commandformat*



The commands are listed in Table 26.7.

Table 26.7 *Commands*

Keyword	*Argument(*s)
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPTTO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RSET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name

Table 26.7 *Commands (continued)*

Keyword	*Argument(s)*
SEND FROM	Intended recipient of the message
SMOLFROM	Intended recipient of the message
SMALFROM	Intended recipient of the message

Responses Responses are sent from the server to the client. A response is a three-digit code that may be followed by additional textual information. Table 26.8 lists some of the responses.

Table 26.8 *Responses*

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted: insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

As the table shows, responses are divided into four categories. The leftmost digit of the code (2, 3, 4, and 5) defines the category.

Mail Transfer Phases

The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.

Example 26.3

Let us see how we can directly use SMTP to send an e-mail and simulate the commands and responses we described in this section. We use TELNET to log into port 25 (the well-known port for SMTP). We then use the commands directly to send an e-mail. In this example, forouzanb@adelphia.net is sending an e-mail to himself. The first few lines show TELNET trying to connect to the Adelphia mail server.

After connection, we can type the SMTP commands and then receive the responses, as shown below. We have shown the commands in black and the responses in color. Note that we have added, for clarification, some comment lines, designated by the "=" signs. These lines are not part of the e-mail procedure.

```
$ telnet mail.adelphia.net 25
Trying 68.168.78.100 •••
Connected to mail.adelphia.net (68.168.78.100).

========= Connection Establishment -- ==========
  220 mta13.adelphia.net SMTP serverready Fri, 6 Aug 2004 . . .
HELO mail.adelphia.net
  250 mta13.adelphia.net
================== Mail Transfer ==============
MAIL FROM: forouzanb@adelphia.net
  250 Sender: <forouzanb@adelphia.net> Ok
RCPT TO: forouzanb@adelphia.net
  250 Recipient <forouzanb@adelphia.net> Ok
DATA
  354 Ok Send data ending with <CRLF>.<CRLF>
From: Forouzan
TO: Forouzan

This is a test message
to show SMTP in action.

========  ========  Connection Termination  ========= =====
  250 Message received: adelphia.net@mail.adelphia.net
QUIT
  221 mta13.adelphia.net SMTP server closing connection
Connection closed by foreign host.
```

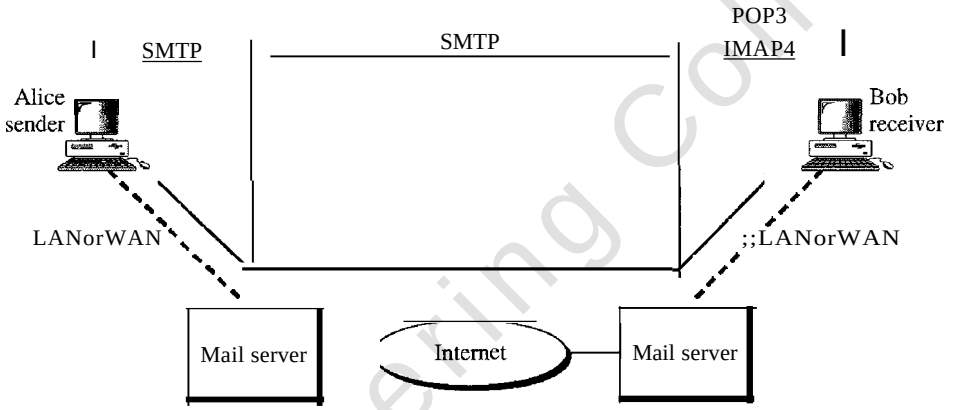
Message Access Agent: **POP** and IMAP

The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from

the client to the server. In other words, the direction of the bulk: data (messages) is from the client to the server. On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. The third stage uses a message access agent.

Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4). Figure 26.19 shows the position of these two protocols in the most common situation (fourth scenario).

Figure 26.19 *POP3 and IMAP4*



POP3

Post Office Protocol, version 3 (POP3) is simple and limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

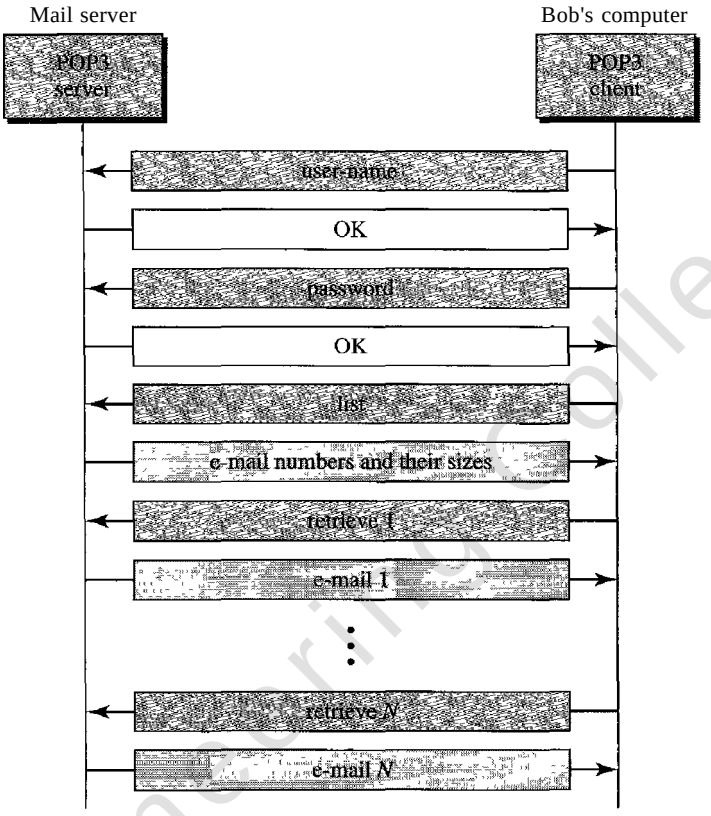
Mail access starts with the client when the user needs to download e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. Figure 26.20 shows an example of downloading using POP3.

POP3 has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying. The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

IMAP4

Another mail access protocol is Internet Mail Access Protocol, version 4 (IMAP4). IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

Figure 26.20 *The exchange of commands and responses in POP3*



POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

Web-Based Mail

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Two common sites are Hotmail and Yahoo. The idea is very simple. Mail transfer from Alice's browser to her mail server is done through HTTP (see Chapter 27). The transfer of the message from the sending mail server to the receiving

mail server is still through SMTP. Finally, the message from the receiving server (the Web server) to Bob's browser is done through HTIP.

The last phase is very interesting. Instead of POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the e-mail is transferred from the Web server to Bob's browser in HTML format.

26.3 FILE TRANSFER

Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to file transfer. In this section, we discuss one popular protocol involved in transferring files: File Transfer Protocol *(FTP).*

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by *TCP/IP* for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first. For example, two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. All these problems have been solved by FTP in a very simple and elegant approach.

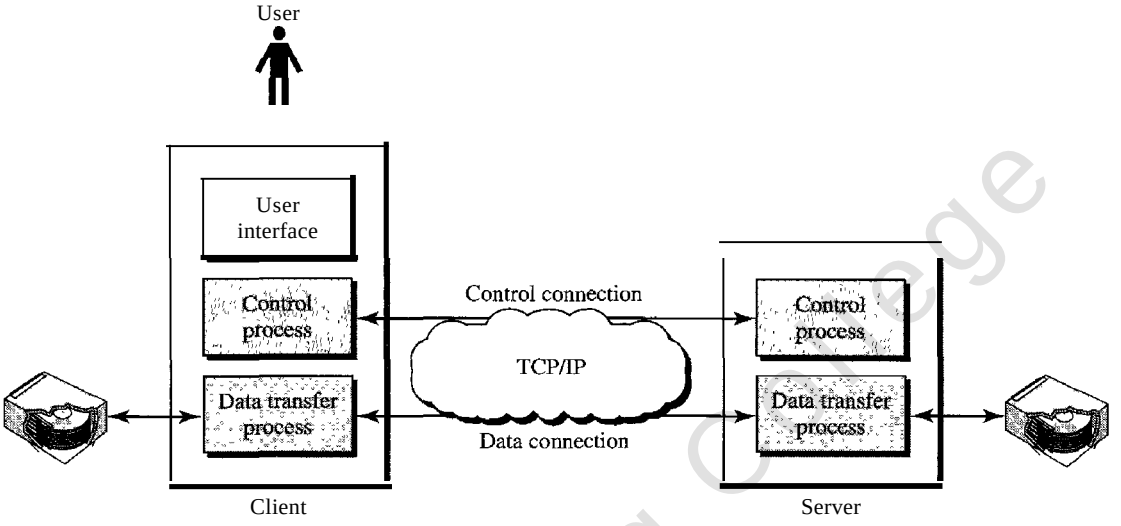
FTP differs from other client/server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. Wc need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred. However, the difference in complexity is at the FTP level, not TCP. For TCP, both connections are treated the same.

FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.

FTP uses the services of TCP. It needs two TCP connections.
The well-known port 21 is used for the control connection
and the well-known port 20 for the data connection.

Figure 26.21 shows the basic model of FTP. The client has three components: user interface, client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.

Figure 26.21 *FTP*

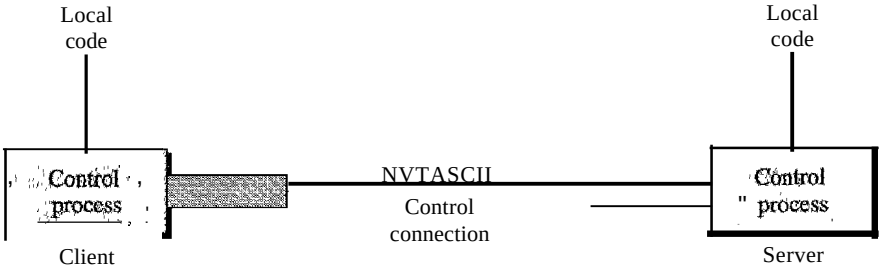


The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transferred. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

Communication over Control Connection

FTP uses the same approach as SMTP to communicate across the control connection. It uses the 7-bit ASCII character set (see Figure 26.22). Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line, so we need not worry about file format or file structure. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.

Figure 26.22 *Using the control connection*



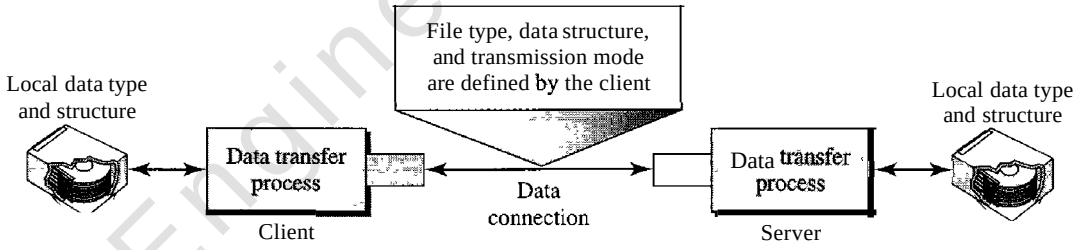
Communication over Data Connection

The purpose of the data connection is different from that of the control connection. We want to transfer files through the data connection. File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things:

- A file is to be copied from the server to the client. This is called *retrieving aft/e*. It is done under the supervision of the RETR command,
- A file is to be copied from the client to the server. This is called *storing aft/e*. It is done under the supervision of the STOR command.
- A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command. Note that FTP treats a list of directory or file names as a file. It is sent over the data connection.

The client must define the type of file to be transferred, the structure of the data, and the transmission mode. Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode (see Figure 26.23).

Figure 26.23 *Using the data connection*



File Type FTP can transfer one of the following file types across the data connection: an ASCII file, EBCDIC file, or image file. The ASCII file is the default format for transferring text files. Each character is encoded using 7-bit ASCII. The sender transforms the file from its own representation into ASCII characters, and the receiver transforms the ASCII characters to its own representation. If one or both ends of the connection use EBCDIC encoding (the file format used by IBM), the file can be transferred using EBCDIC encoding. The image file is the default format for transferring binary files. The file is sent as continuous streams of bits without any interpretation or encoding. This is mostly used to transfer binary files such as compiled programs.

Data Structure FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: file structure, record structure, and page structure. In the file structure format, the file is a continuous stream of bytes. In the record structure, the file is divided into records. This can be used only with text files. In the page structure, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Transmission Mode FTP can transfer a file across the data connection by using one of the following three transmission modes: stream mode, block mode, and compressed mode. The stream mode is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data are simply a stream of bytes (file structure), no end-of-file is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a I-byte end-of-record (EOR) character and the end of the file will have a I-byte end-of-file (EOF) character. In block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the *block descriptor;* the next 2 bytes define the size of the block in bytes. In the compressed mode, if the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the number of repetitions. In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed.

Example 26.4

The following shows an actual *FTP* session for retrieving a list of items in a directory. The colored lines show the responses from the server control connection; the black lines show the commands sent by the client. The lines in white with a black background show data transfer.

```
$ ftp voyager.deanza.tbda.edu
Connected to voyager.deanza.tbda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.tbda.edu:forouzan): forouzan
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls reports
227 Entering Passive Mode (153,18,17,11,238,169)
150 Here comes the directory listing.
```

```
226 Directory send OK.
ftp>quit
221 Goodbye.
```

1. After the control connection is created, the FIP server sends the 220 (service ready) response on the control connection.
2. The client sends its name.
3. The server responds with 331 (user name is OK, password is required).

4. The client sends the password (not shown).
5. The server responds with 230 (user log-in is OK).
6. The client sends the list command Os reports) to find the list of files on the directory named report.
7. Now the server responds with 150 and opens the data connection.
8. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
9. The client now has two choices. It can use the QUIT command to request the closing of the control connection, or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
10. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.

Anonymous FTP

To use FfP, a user needs an account (user name) and a password on the remote server. Some sites have a set of files available for public access, to enable anonymous FTP. To access these files, a user does not need to have an account or password. Instead, the user can use *anonymous* as the user name and *guest* as the password.

User access to the system is very limited. Some sites allow anonymous users only a subset of commands. For example, most sites allow the user to copy some files, but do not allow navigation through the directories.

Example 26.5

We show an example of anonymous FTP. We assume that some public data are available at intemic.net.

```
$ ftp internic.net
Connected to internic.net
220 Server ready
Name: anonymous
331 Guest login OK, send "guest" as password
Password: guest
ftp>pwd"
257 '/' is current directory
ftp > ls
200 OK
150 Opening ASCII mode
```

```
ttp> close
221 Goodbye
ftp>quit
```

26.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

Remote logging is discussed in Chapter 18 of [For06] and Chapter 26 of [Ste94]. Electronic mail is discussed in Chapter 20 of [For06], Section 9.2 of [PD03], Chapter 32 of [Com04], Section 7.2 of [Tan03], and Chapter 28 of [Ste94]. FTP is discussed in Chapter 19 of [For06], Chapter 27 of [Ste94], and Chapter 34 of [Com04].

Sites

The following sites are related to topics discussed in this chapter.

- www.ietf.org/rfc.htrnl Information about RFCs

RFCs

The following RFCs are related to TELNET:

137, 340, 393, 426, 435, 452, 466, 495, 513, 529, 562, 595, 596, 599, 669, 679, 701, 702, 703, 728, 764, 782, 818, 854, 855, 1184, 1205, 2355

The following RFCs are related to SMTP, POP, and IMAP:

196, 221, 224, 278, 524, 539, 753, 772, 780, 806, 821, 934, 974, 1047, 1081, 1082, 1225, 1460, 1496, 1426, 1427, 1652, ..., 1711, 1725, 1734, 1740, 1741, 1767, 1869, 1870, 2045, 2046, 2047, 2048, 2177, 2180, 2192, 2193, 2221, 2342, 2359, 2449, 2683, 2503

The following RFCs are related to FTP:

114, 133, 141, 163, 171, 172, 238, 242, 250, 256, 264, 269, 281, 291, 354, 385, 412, 414, 418, 430, 438, 448, 463, 468, 478, 486, 505, 506, 542, 553, 624, 630, 640, 691, 765, 913, 959, 1635, 1785, 2228, 2577

DNS is discussed in [AL98], chapter 17 of [For06], section 9.1 of [PD03], and section 7.1 of [Tan03].

26.5 KEY TERMS

alias
anonymous FTP
ASCII file
block mode
body
character mode
compressed mode
control character
control connection
data connection

default mode
domain name
envelope
file structure
File Transfer Protocol (FTP)
header
image file
Internet Mail Access Protocol, version 4 (IMAP4)
line mode
local log-in
local part
message access agent (MAA)
message transfer agent (MTA)
Multipurpose Internet Mail Extensions (MIME)
network virtual terminal (NVT)
option negotiation
page structure
Post Office Protocol, version 3 (POP3)
record structure
remote log-in
Simple Mail Transfer Protocol (SMTP)
stream mode
suboption negotiation
terminal network (TELNET)
timesharing
user agent *(VA)*

26.6 SUMMARY

- TELNET is a client/server application that allows a user to log on to a remote machine, giving the user access to the remote system.
- TELNET uses the network virtual terminal (NVT) system to encode characters on the local system. On the server machine, NVT decodes the characters to a form acceptable to the remote machine.
- NVT uses a set of characters for data and a set of characters for control.
- In TELNET, control characters are embedded in the data stream and preceded by the *interpret as control* (lAC) control character.
- Options are features that enhance the TELNET process.
- TELNET allows negotiation to set transfer conditions between the client and server before and during the use of the service.
- A TELNET implementation operates in the default, character, or line mode.
  1. In the default mode, the client sends one line at a time to the server.
  2. In the character mode, the client sends one character at a time to the server.
  3. In the line mode, the client sends one line at a time to the server.
- Several programs, including SMTP, POP3, and IMAP4, are used in the Internet to provide electronic mail services.
- In electronic mail, the *VA* prepares the message, creates the envelope, and puts the message in the envelope.
- In electronic mail, the mail address consists of two parts: a local part (user mailbox) and a domain name. The form is localpart@domainname.
- In electronic mail, Multipurpose Internet Mail Extension (MIME) allows the transfer of multimedia messages.
- In electronic mail, the MTA transfers the mail across the Internet, a LAN, or a WAN.

- SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.
- The steps in transferring a mail message are
  1. Connection establishment·
  2. Mail transfer
  3. Connection termination
- Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4) are protocols used for pulling messages from a mail server.
- One of the programs used for file transfer in the Internet is File Transfer Protocol (FTP).
- FTP requires two connections for data transfer: a control connection and a data connection.
- FTP employs NVT ASCII for communication between dissimilar systems.
- Prior to the actual transfer of files, the file type, data structure, and transmission mode are defined by the client through the control connection.
- Responses are sent from the server to the client during connection establishment.
- There are three types of file transfer:
  1. A file is copied from the server to the client.
  2. A file is copied from the client to the server.
  3. A list of directories or file names is sent from the server to the client.
- Anonymous FTP provides a method for the general public to access files on remote sites.

26.7 PRACTICE SET

Review Questions

1. What is the difference between local and remote log-in in TELNET?
2. How are control and data characters distinguished in NVT?
3. How are options negotiated in TELNET?
4. Describe the addressing system used by SMTP.
5. In electronic mail, what are the tasks of a user agent?
6. In electronic mail, what is MIME?
7. Why do we need POP3 or IMAP4 for electronic mail?
8. What is the purpose of FTP?
9. Describe the functions of the two FTP connections.
10. What kinds of file types can FTP transfer?
11. What are the three FTP transmission modes?
12. How does storing a file differ from retrieving a file?
13. What is anonymous FTP?

Exercises

14. Show the sequence of bits sent from a client TELNET for the binary transmission of11110011 00111100 11111111.
15. If TELNET is using the character mode, how many characters are sent back and forth between the client and server to copy a file named file1 to another file named file2 using the command *cp file1 file2?*
16. What is the minimum number of bits sent at the TCP level to accomplish the task in Exercise *IS?*
17. What is the minimum number of bits sent at the data link layer level (using Ethernet) to accomplish the task in Exercise 15?
18. What is the ratio of the useful bits to the total bits in Exercise 17?
19. Interpret the following sequences of characters (in hexadecimal) received by a TELNET client or server.
    a. FFFB 01
    b. FFFEOI
    c. FFF4
    d. FFF9
20. A sender sends unformatted text. Show the MIME header.
21. A sender sends a IPEG message. Show the MIME header.
22. Why is a connection establishment for mail transfer needed if TCP has already established a connection?
23. Why should there be limitations on anonymous FTP? What could an unscrupulous user do?
24. Explain why FTP does not have a message format.

Research Activities

25. Show the sequence of characters exchanged between the TELNET client and the server to switch from the default mode to the character mode.
26. Show the sequence of characters exchanged between the TELNET client and the server to switch from the default mode to line mode.
27. In SMTP, show the connection establishment phase from aaa@xxx.com to bbb@yyy.com.
28. In SMTP, show the message transfer phase from aaa@xxx.com to bbb@yyy.com. The message is "Good morning my friend."
29. In SMTP, show the connection termination phase from aaa@xxx.com to bbb@yyy.com.
30. What do you think would happen if the control connection were accidentally severed during an FTP transfer?

31. Find the extended options proposed for TELNET.
32. Another log-in protocol is called Rlogin. Find some information about Rlogin and compare it with TELNET.
33. A more secure log-in protocol in UNIX is called Secure Shell (SSH). Find some information about this protocol.

CHAPTER 27

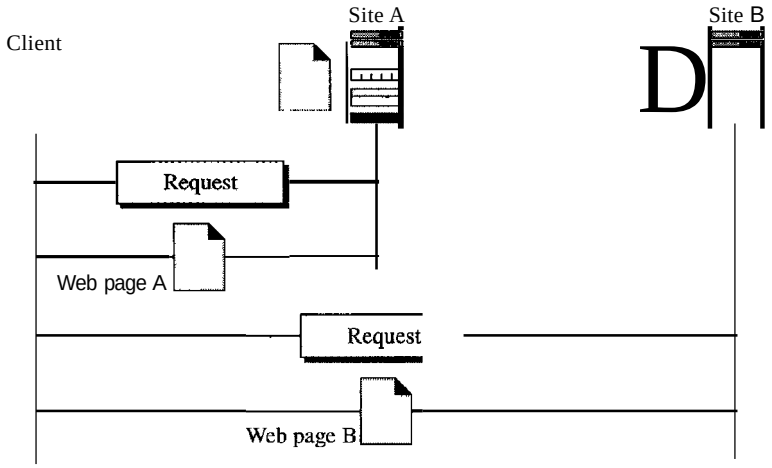
WWWandHTTP

The **World Wide Web** (WWW) is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research. **In** this chapter we first discuss issues related to the Web. We then discuss a protocol, HTTP, that is used to retrieve information from the Web.

27.1 ARCHITECTURE

The WWW today is a distributed clientJserver service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites,* as shown **in** Figure 27.1.

Figure 27.1 *Architecture ofWWW*

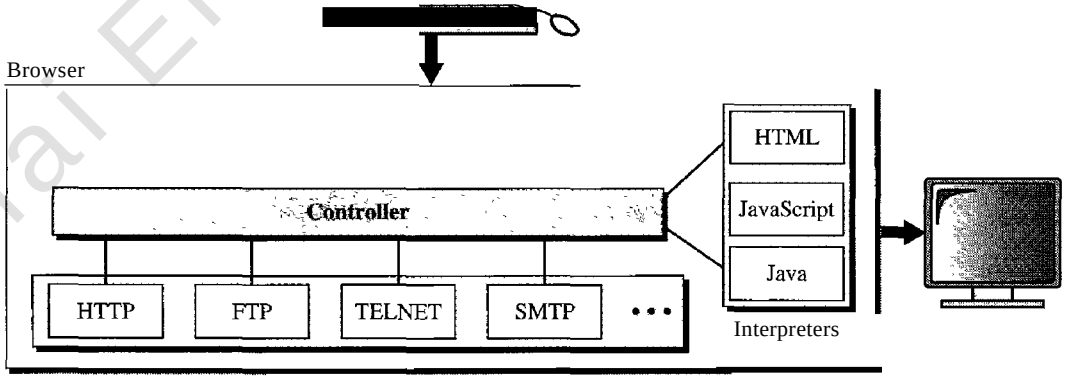


Each site holds one or more documents, referred to as *Web pages*. Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers. Let us go through the scenario shown in Figure 27.1. The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL, which we will discuss shortly. The server at site A finds the document and sends it to the client. When the user views the document, she finds some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as FTP or HTIP (described later in the chapter). The interpreter can be HTML, Java, or JavaScript, depending on the type of document. We discuss the use of these interpreters based on the document type later in the chapter (see Figure 27.2).

Figure 27.2 *Browser*



Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time.

Uniform Resource Locator

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path (see Figure 27.3).

Figure 27.3 *URL*

The *protocol* is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

Cookies

The World Wide Web was originally designed as a stateless entity. A client sends a request; a server responds. Their relationship is over. The original design of WWW, retrieving publicly available documents, exactly fits this purpose. Today the Web has other functions; some are listed here.

1. Some websites need to allow access to registered clients only.
2. Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
3. Some websites are used as portals: the user selects the Web pages he wants to see.
4. Some websites are just advertising.

For these purposes, the cookie mechanism was devised. We discussed the use of cookies at the transport layer in Chapter 23; we now discuss their use in Web pages.

Creation and Storage of Cookies

The creation and storage of cookies depend on the implementation; however, the principle is the same.

1. When a server receives a request from a client, it stores information about the client in a file or a string. The information may include the domain name of the client, the contents of the cookie (information the server has gathered about the client such as name, registration number, and so on), a timestamp, and other information'depending on the implementation.
2. The server includes the cookie in the response that it sends to the client.
3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the domain server name.

Using Cookies

When a client sends a request to a server, the browser looks in the cookie directory to see if it can find a cookie sent by that server. If found, the cookie is included in the request. When the server receives the request, it knows that this is an old client, not a new one. Note that the contents of the cookie are never read by the browser or disclosed to the user. It is a cookie *made* by the server and *eaten* by the server. Now let us see how a cookie is used for the four previously mentioned purposes:

1. The site that restricts access to registered clients only sends a cookie to the client when the client registers for the first time. For any repeated access, only those clients that send the appropriate cookie are allowed.
2. An electronic store (e-commerce) can use a cookie for its client shoppers. When a client selects an item and inserts it into a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser. If the client selects a second item, the cookie is updated with the new selection information. And so on. When the client finishes shopping and wants to check out, the last cookie is retrieved and the total charge is calculated.
3. A Web portal uses the cookie in a similar way. When a user selects her favorite pages, a cookie is made and sent. If the site is accessed again, the cookie is sent to the server to show what the client is looking for.
4. A cookie is also used by advertising agencies. An advertising agency can place banner ads on some main website that is often visited by users. The advertising agency supplies only a URL that gives the banner address instead of the banner itself. When a user visits the main website and clicks on the icon of an advertised corporation, a request is sent to the advertising agency. The advertising agency sends the banner, a GIF file, for example, but it also includes a cookie with the ill of the user. Any future use of the banners adds to the database that profiles the Web behavior of the user. The advertising agency has compiled the interests of the user and can sell this information to other parties. This use of cookies has made them very controversial. Hopefully, some new regulations will be devised to preserve the privacy of users.

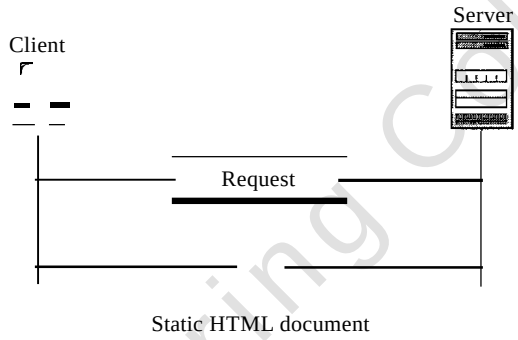
27.2 WEB DOCUMENTS

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active. The category is based on the time at which the contents of the document are determined.

Static Documents

Static documents are fixed-content documents that are created and stored in a server. The client can get only a copy of the document. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, the contents in the server can be changed, but the user cannot change them. When a client accesses the document, a copy of the document is sent. The user can then use a browsing program to display the document (see Figure 27.4).

Figure 27.4 *Static document*

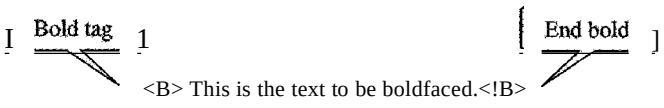


HTML

Hypertext Markup Language (HTML) is a language for creating Web pages. The term *markup language* comes from the book publishing industry. Before a book is typeset and printed, a copy editor reads the manuscript and puts marks on it. These marks tell the compositor how to format the text. For example, if the copy editor wants part of a line to be printed in boldface, he or she draws a wavy line under that part. In the same way, data for a Web page are formatted for interpretation by a browser.

Let us clarify the idea with an example. To make part of a text displayed in boldface with HTML, we put beginning and ending boldface tags (marks) in the text, as shown in Figure 27.5.

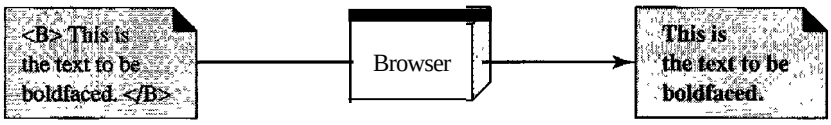
Figure 27.5 *Boldface tags*



The two tags <B> and </B> are instructions for the browser. When the browser sees these two marks, it knows that the text must be boldfaced (see Figure 27.6).

A markup language such as HTML allows us to embed formatting instructions in the file itself. The instructions are included with the text. In this way, any browser can read the instructions and format the text according to the specific workstation. One might

Figure 27.6 *Effect of boldface tags*



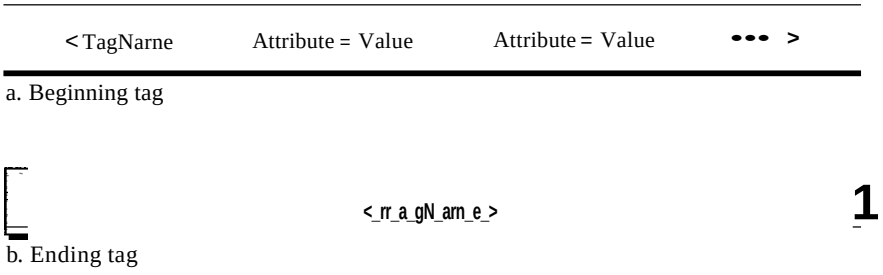
ask why we do not use the fonnatting capabilities of word processors to create and save formatted text. The answer is that different word processors use different techniques or procedures for formatting text. For example, imagine that a user creates formatted text on a Macintosh computer and stores it in a Web page. Another user who is on an IBM computer would not be able to receive the Web page because the two computers use different fonnatting procedures.

HTML lets us use only ASCII characters for both the main text and formatting instructions. In this way, every computer can receive the whole document as an ASCII document. The main text is the data, and the formatting instructions can be used by the browser to format the data.

A Web page is made up of two parts: the head and the body. The head is the first part of a Web page. The head contains the title of the page and other parameters that the browser will use. The actual contents of a page are in the body, which includes the text and the tags. Whereas the text is the actual infonnation contained in a page, the tags define the appearance of the document. Every HTML tag is a name followed by an optional list of attributes, all enclosed between less-than and greater-than symbols « and >).

An attribute, if present, is followed by an equals sign and the value of the attribute. Some tags can be used alone; others must be used in pairs. Those that are used in pairs are called *beginning* and *ending* tags. The beginning tag can have attributes and values and starts with the name of the tag. The ending tag cannot have attributes or values but must have a slash before the name of the tag. The browser makes a decision about the structure of the text based on the tags, which are embedded into the text. Figure 27.7 shows the fonnat of a tag.

Figure 27.7 *Beginning and ending tags*



One commonly used tag category is the text formatting tags such as <B> and <*/B*>, which make the text bold; <1> and <II>, which make the text italic; and <U> and <*IV*>, which underline the text.

Another interesting tag category is the image tag. Nontextual information such as digitized photos or graphic images is not a physical part of an HTML document. But we can use an image tag to point to the file of a photo or image. The image tag defines the address (URL) of the image to be retrieved. It also specifies how the image can be inserted after retrievaL We can choose from several attributes. The most common are SRC (source), which defines the source (address), and ALIGN, which defines the alignment of the image. The SRC attribute is required. Most browsers accept images in the GIF or IPEG formats. For example, the following tag can retrieve an image stored as imagel.gif in the directory /bin/images:

```
<IMG SRC= "/bin/images/image1.gif" ALIGN=MIDDLE >
```

A third interesting category is the hyperlink tag, which is needed to link documents together. Any item (word, phrase, paragraph, or image) can refer to another document through a mechanism called an *anchor.* The anchor is defined by <A ... > and <!A> tags, and the anchored item uses the URL to refer to another document. When the document is displayed, the anchored item is underlined, blinking, or boldfaced. The user can click on the anchored item to go to another document, which mayor may not be stored on the same server as the original document. The reference phrase is embedded between the beginning and ending tags. The beginning tag can have several attributes, but the one required is HREF (hyperlink reference), which defines the address (URL) of the linked document. For example, the link to the author of a book can be

```
<A HREF= "http://www.deanza.edu/forouzan"> Author </A>
```

What appears in the text is the word *Author,* on which the user can click to go to the author's Web page.

Dynamic Documents

A **dynamic document** is created by a Web server whenever a browser requests the document. When a request arrives, the Web server runs an application program or a script that creates the dynamic document. The server returns the output of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document can vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server. Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the *date* program in UNIX and send the result of the program to the client.

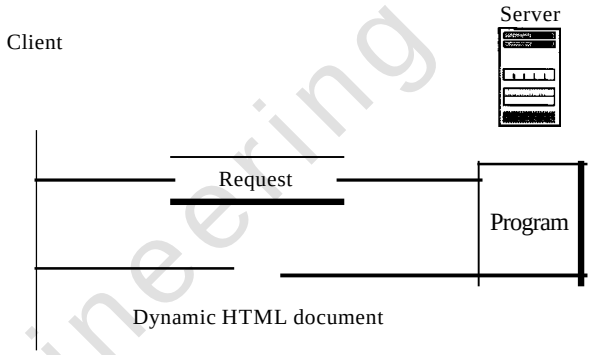
Common Gateway Interface (CGI)

The **Common** Gateway **Interface** (CGI) is a technology that creates and handles dynamic documents. CGI is a set of standards that defines how a dynamic document is written, how data are input to the program, and how the output result is used.

COl is not a new language; instead, it allows programmers to use any of several languages such as C, C++, Boume Shell, Kom Shell, C Shell, Tcl, or Perl. The only thing that CGI defines is a set of rules and tenns that the programmer must follow.

The tenn *common* in COl indicates that the standard defines a set of rules that is common to any language or platfonn. The tenn *gateway* here means that a COl program can be used to access other resources such as databases, graphical packages, and so on. The tenn *interface* here means that there is a set of predefined tenns, variables, calls, and so on that can be used in any COl program. A COl program in its simplest fonn is code written in one of the languages supporting COL Any programmer who can encode a sequence of thoughts in a program and knows the syntax of one of the above-mentioned languages can write a simple CGI program. Figure 27.8 illustrates the steps in creating a dynamic program using COl technology.

Figure 27.8 *Dynamic document using CGI*



Input In traditional programming, when a program is executed, parameters can be passed to the program. Parameter passing allows the programmer to write a generic program that can be used in different situations. For example, a generic copy program can be written to copy any file to another. A user can use the program to copy a file named *x* to another file named *y* by passing *x* and *y* as parameters.

The input from a browser to a server is sent by using *aform*. If the infonnation in a fonn is small (such as a word), it can be appended to the URL after a question mark. For example, the following URL is carrying fonn infonnation (23, a value):

http://www.deanzalcgi-binlprog.pl?23

When the server receives the URL, it uses the part of the URL before the question mark to access the program to be run, and it interprets the part after the question mark (23) as the input sent by the client. It stores this string in a variable. When the CGI program is executed, it can access this value.

If the input from a browser is too long to fit in the query string, the browser can ask the server to send a fonn. The browser can then fill the fonn with the input data and send it to the server. The infonnation in the fonn can be used as the input to the COl program.

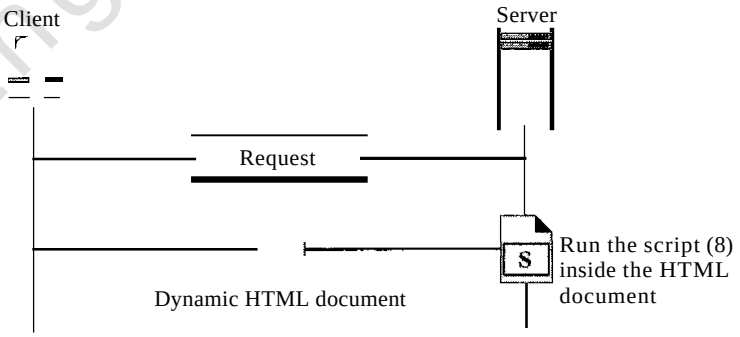
Output The whole idea of CGI is to execute a CGI program at the server site and send the output to the client (browser). The output is usually plain text or a text with HTML structures; however, the output can be a variety of other things. It can be graphics or binary data, a status code, instructions to the browser to cache the result, or instructions to the server to send an existing document instead of the actual output.

To let the client know about the type of document sent, a CGI program creates headers. As a matter of fact, the output of the CGI program always consists of two parts: a header and a body. The header is separated by a blank line from the body. This means any CGI program creates first the header, then a blank line, and then the body. Although the header and the blank line are not shown on the browser screen, the header is used by the browser to interpret the body.

Scripting Technologies for Dynamic Documents

The problem with CGI technology is the inefficiency that results if part of the dynamic document that is to be created is fixed and not changing from request to request. For example, assume that we need to retrieve a list of spare parts, their availability, and prices for a specific car brand. Although the availability and prices vary from time to time, the name, description, and the picture of the parts are fixed. If we use CGI, the program must create an entire document each time a request is made. The solution is to create a file containing the fixed part of the document using HTML and embed a script, a source code, that can be run by the server to provide the varying availability and price section. Figure 27.9 shows the idea.

Figure 27.9 *Dynamic document using server-site script*



A few technologies have been involved in creating dynamic documents using scripts. Among the most common are Hypertext Preprocessor (pHP), which uses the Perl language; Java Server Pages (JSP), which uses the Java language for scripting; Active Server Pages (ASP), a Microsoft product which uses Visual Basic language for scripting; and ColdFusion, which embeds SQL database queries in the HTML document.

Dynamic documents are sometimes referred to as server-site dynamic documents.

Active Documents

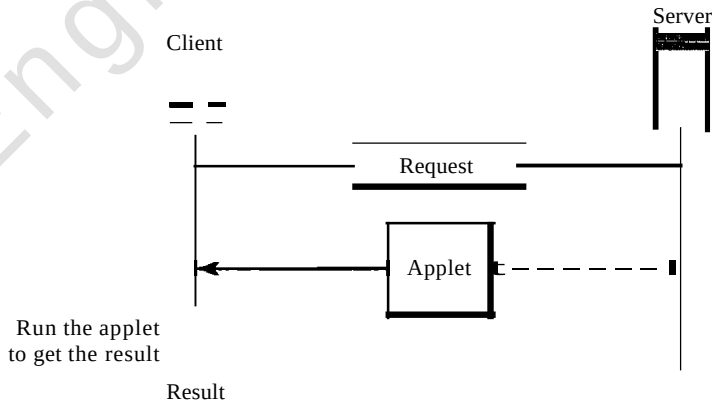
For many applications, we need a program or a script to be run at the client site. These are called active documents. For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place. When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client (browser) site.

Java Applets

One way to create an active document is to use Java applets. Java is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document (an applet) and a browser to run it. It can also be a stand-alone program that doesn't use a browser.

An applet is a program written in Java on the server. It is compiled and ready to be run. The document is in byte-code (binary) format. The client process (browser) creates an instance of this applet and runs it. A Java applet can be run by the browser in two ways. In the first method, the browser can directly request the Java applet program in the URL and receive the applet in binary form. In the second method, the browser can retrieve and run an HTML file that has embedded the address of the applet as a tag. Figure 27.10 shows how Java applets are used in the first method; the second is similar but needs two transactions.

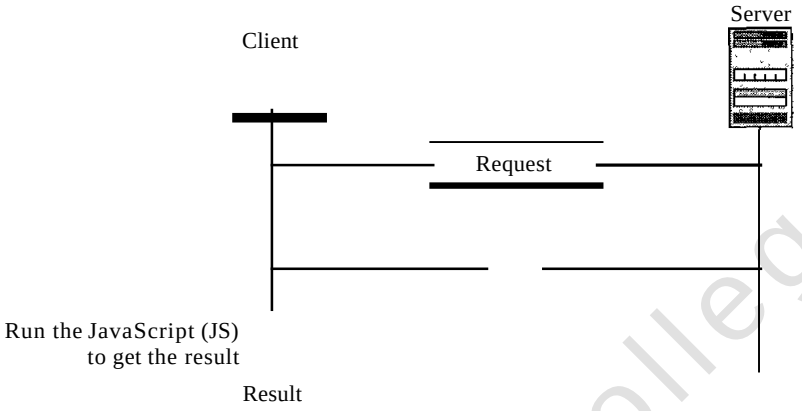
Figure 27.10 *Active document using Java applet*



JavaScript

The idea of scripts in dynamic documents can also be used for active documents. If the active part of the document is small, it can be written in a scripting language; then it can be interpreted and run by the client at the same time. The script is in source code (text) and not in binary form. The scripting technology used in this case is usually JavaScript. JavaScript, which bears a small resemblance to Java, is a very high level scripting language developed for this purpose. Figure 27.11 shows how JavaScript is used to create an active document.

Figure 27.11 *Active document using client-site script*



Active documents are sometimes referred to as client-site dynamic documents.

27.3 HTTP

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. It is similar to FfP because it transfers files and uses the services of TCP. However, it is much simpler than FfP because it uses only one TCP connection. There is no separate control connection; only data are transferred between the client and the server.

HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP uses the services of TCP on well-known port 80.

HTTP Transaction

Figure 27.12 illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.

Messages

The formats of the request and response messages are similar; both are shown in Figure 27.13. A request message consists of a request line, a header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body.

Figure 27.12 *HTTP transaction*

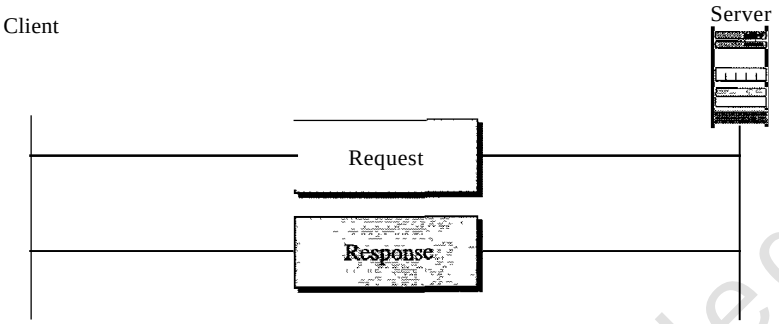
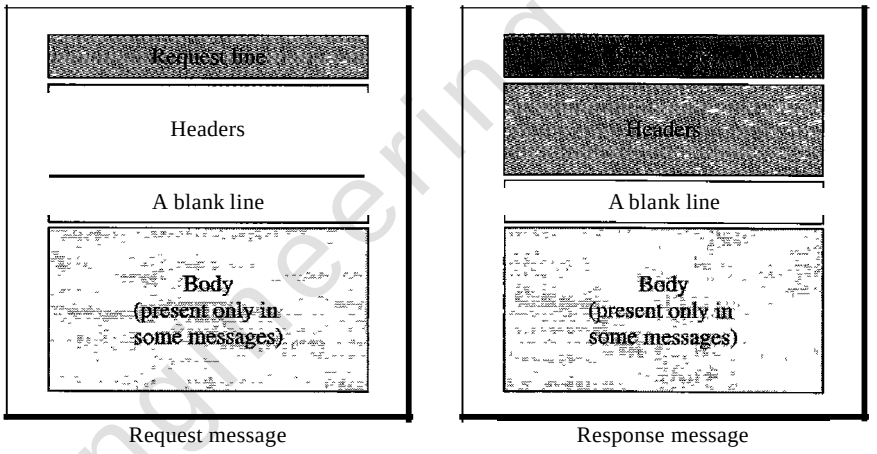
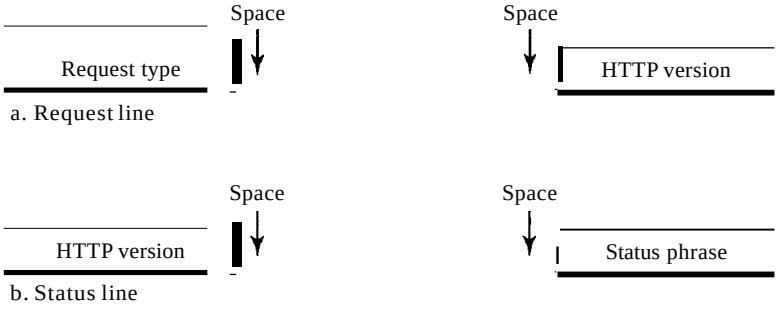


Figure 27.13 *Request and response messages*



Request and Status Lines The first line in a request message is called a request line; the first line in the response message is called the status line. There is one common field, as shown in Figure 27.14.

Figure 27.14 *Request and status lines*



- Request type. This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The request type is categorized into *methods* as defined in Table 27.1.

Table 27.1 *Methods*

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

- URL. We discussed the URL earlier in the chapter.
- Version. The most current version of HTTP is 1.1.
- Status code. This field is used in the response message. The status code field is similar to those in the FTP and the SMTP protocols. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site. We list the most common codes in Table 27.2.
- Status phrase. This field is used in the response message. It explains the status code in text form. Table 27.2 also gives the status phrase.

Table 27.2 *Status codes*

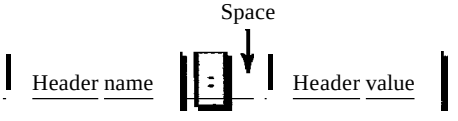
Code	*Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Table 27.2 *Status codes (continued)*

Code	*Phrase*	*Description*
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Header The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value (see Figure 27.15). We will show some header lines in the examples at the end of this chapter. A header line belongs to one of four categories: general header, request header, response header, and entity header. A request message can contain only general, request, and entity headers. A response message, on the other hand, can contain only general, response, and entity headers.

Figure 27.15 *Header format*



- General header The general header gives general information about the message and can be present in both a request and a response. Table 27.3 lists some general headers with their descriptions.

Table 27.3 *General headers*

Header	*Description*
Cache-control	Specifies infonnation about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

- Request header The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format. See Table 27.4 for a list of some request headers and their descriptions.

Table 27.4 *Request headers*

Header	*Description*
Accept	Shows the medium fonnat the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what pennissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

- Response header The response header can be present only in a response message. It specifies the server's configuration and special information about the request. See Table 27.5 for a list of some response headers with their descriptions.

Table 27.5 *Response headers*

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

- Entity header The entity header gives infonnation about the body of the document. Although it is mostly present in response messages, some request messages, such as POST or PUT methods, that contain a body also use this type of header. See Table 27.6 for a list of some entity headers and their descriptions.

Table 27.6 *Entity headers*

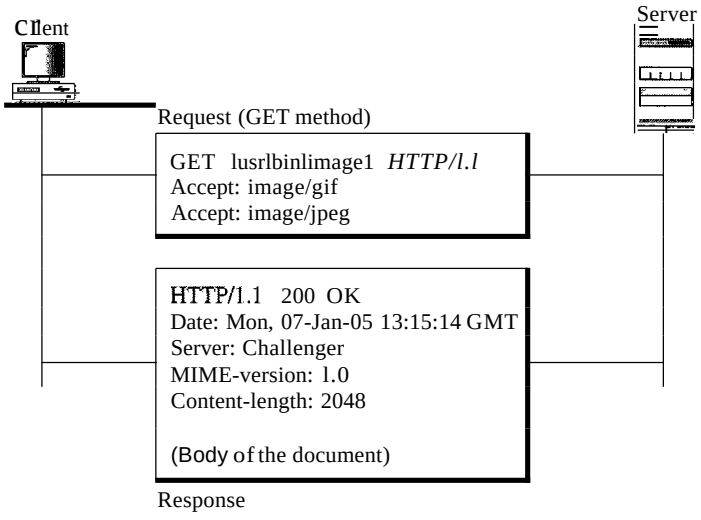
Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Body The body can be present in a request or response message. Usually, it contains the document to be sent or received.

Example 27.1

This example retrieves a document. We use the GET method to retrieve an image with the path /usr/bin/imagel. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body. The response message contains the status line and four lines of header. The header lines define the date, server, MIME version, and length of the document. The body of the document follows the header (see Figure 27.16).

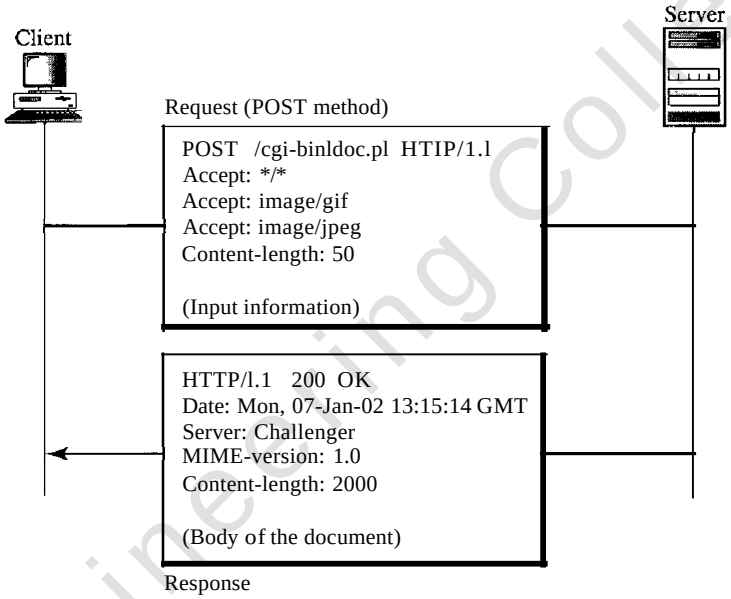
Figure 27.16 *Example 27.1*



Example 27.2

In this example, the client wants to send data to the server. We use the POST method. The request line shows the method (POST), URL, and HTTP version (1.1). There are four lines of headers. The request body contains the input information. The response message contains the status line and four lines of headers. The created document, which is a CGI document, is included as the body (see Figure 27.17).

Figure 27.17 *Example 27.2*



Example 27.3

HTTP uses ASCII characters. A client can directly connect to a server using TELNET, which logs into port 80. The next three lines show that the connection is successful.

We then type three lines. The first shows the request line (GET method), the second is the header (defining the host), the third is a blank, terminating the request.

The server response is seven lines starting with the status line. The blank line at the end terminates the server response. The file of 14,230 lines is received after the blank line (not shown here). The last line is the output by the client.

```
$ teinet www.mhhe.com 80
Trying 198.45.24.104 ...
Connected to www.mhhe.com (198.45.24.104).
Escape character is 11\]'.
GET /engcslcompscilforouzan HTTP/I.t
From: forouzanbehrouz@fbda.edu

HTTP/t.l 200 OK
Date: Thu, 28 Oct 2004 16:27:46 GMT
Server: Apache/l.3.9 (Unix) ApacheJServ/1.1.2 PHP/4.1.2 PHP/3.0.18
MIME-version: 1.0
Content-Type: text/html
```

```
Last-modified: Friday, 15-0ct-04 02:11:31 GMT
Content-length: 14230

Connection closed by foreign host.
```

Persistent Versus Nonpersistent Connection

HTTP prior to version 1.1 specified a nonpersistent connection, while a persistent connection is the default in version 1.1.

Nonpersistent Connection

In a nonpersistent connection, one TCP connection is made for each request/response. The following lists the steps in this strategy:

1. The client opens a TCP connection and sends a request.
2. The server sends the response and closes the connection.
3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

In this strategy, for N different pictures in different files, the connection must be opened and closed N times. The nonpersistent strategy imposes high overhead on the server because the server needs N different buffers and requires a slow start procedure each time a connection is opened.

Persistent Connection

HTTP version 1.1 specifies a persistent connection by default. In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response. However, there are some occasions when the sender does not know the length of the data. This is the case when a document is created dynamically or actively. In these cases, the server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached.

HTTP version 1.1 specifies a persistent connection by default.

Proxy Server

HTTP supports proxy servers. A proxy server is a computer that keeps copies of responses to recent requests. The HTTP client sends a request to the proxy server. The proxy server checks its cache. If the response is not stored in the cache, the proxy server sends the request to the corresponding server. Incoming responses are sent to the proxy server and stored for future requests from other clients.

The proxy server reduces the load on the original server, decreases traffic, and improves latency. However, to use the proxy server, the client must be configured to access the proxy instead of the target server.

27.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

HTTP is discussed in Chapters 13 and 14 of [Ste96], Section 9.3 of [PD03], Chapter 35 of [Com04], and Section 7.3 of [Tan03].

Sites

The following sites are related to topics discussed in this chapter.

- www.ietf.org/rfc.html Information about RFCs

RFCs

The following RFCs are related to WWW:

1614,1630,1737,1738

The following RFCs are related to HTTP:

2068,2109

27.5 KEY TERMS

active document
Active Server Pages (ASP)
applet
browser
ColdFusion
Common Gateway Interface (CGI)
dynamic document
entity header
general header
host
Hypertext Markup Language (HTML)
Hypertext Preprocessor (PHP)
Hypertext Transfer Protocol (HTTP)
Java
JavaScript
Java Server Pages (JSP)
nonpersistent connection
path
persistent connection
proxy server
request header
request line
request type
response header
static document
status code
status line
tag
uniform resource locator (URL)
Web
World Wide Web (WWW)

27.6 SUMMARY

- The World Wide Web (WWW) is a repository of information linked together from points all over the world.
- Hypertexts are documents linked to one another through the concept of pointers.
- Browsers interpret and display a Web document.
- A browser consists of a controller, client programs, and interpreters.
- A Web document can be classified as static, dynamic, or active.
- A static document is one in which the contents are fixed and stored in a server. The client can make no changes in the server document.
- Hypertext Markup Language (HTML) is a language used to create static Web pages.
- Any browser can read formatting instructions (tags) embedded in an HTML document.
- Tags provide structure to a document, define titles and headers, format text, control the data flow, insert figures, link different documents together, and define executable code.
- A dynamic Web document is created by a server only at a browser request.
- The Common Gateway Interface (CGI) is a standard for creating and handling dynamic Web documents.
- A CGI program with its embedded CGI interface tags can be written in a language such as C, C++, Shell Script, or Perl.
- An active document is a copy of a program retrieved by the client and run at the client site.
- Java is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document and a browser to run it.
- Java is used to create applets (small application programs).
- The Hypertext Transfer Protocol (HTTP) is the main protocol used to access data on the World Wide Web (WWW).
- HTTP uses a TCP connection to transfer files.
- An HTTP message is similar in form to an SMTP message.
- The HTTP request line consists of a request type, a URL, and the HTTP version number.
- The uniform resource locator (URL) consists of a method, host computer, optional port number, and path name to locate information on the WWW.
- The HTTP request type or method is the actual command or request issued by the client to the server.
- The status line consists of the HTTP version number, a status code, and a status phrase.
- The HTTP status code relays general information, information related to a successful request, redirection information, or error information.
- The HTTP header relays additional information between the client and server.

- An HTTP header consists of a header name and a header value.
- An HTTP general header gives general information about the request or response message.
- An HTTP request header specifies a client's configuration and preferred document format.
- An HTTP response header specifies a server's configuration and special information about the request.
- An HTTP entity header provides information about the body of a document.
- HTTP, version 1.1, specifies a persistent connection.
- A proxy server keeps copies of responses to recent requests.

27.7 PRACTICE SET

Review Questions

1. How is HTTP related to WWW?
2. How is HTTP similar to SMTP?
3. How is HTTP similar to *FTP?*
4. What is a URL and what are its components?
5. What is a proxy server and how is it related to HTTP?
6. Name the common three components of a browser.
7. What are the three types of Web documents?
8. What does HTML stand for and what is its function?
9. What is the difference between an active document and a dynamic document?
10. What does CGI stand for and what is its function?
11. Describe the relationship between Java and an active document.

Exercises

12. Where will each figure be shown on the screen?

```
Look at the following picture:
then tell me what you feel:
<IMG SRC="PictureslFunnyl.gif" ALIGN=middle>
<lMG SRC="Pictures/Funny2.gif" ALIGN=bottom>
<B>What is your feeling? <IE>
```

13. Show the effect of the following HTML segment.

```
The publisher of this book is <A HREF="www.mhhe">
McGraw-Hill Publisher <fA>
```

14. Show a request that retrieves the document /usr/users/doc/docl. Use at least two general headers, two request headers, and one entity header.
15. Show the response to Exercise 14 for a successful request.
16. Show the response to Exercise 14 for a document that has permanently moved to /usr/deads/doc1.

17. Show the response to Exercise 14 if there is a syntax error in the request.
18. Show the response to Exercise 14 if the client is unauthorized to access the document.
19. Show a request that asks for information about a document at /bin/users/file. Use at least two general headers and one request header.
20. Show the response to Exercise 19 for a successful request.
21. Show the request to copy the file at location /bin/usr/bin/file1 to /bin/file1.
22. Show the response to Exercise 21.
23. Show the request to delete the file at location /bin/file!.
24. Show the response to Exercise 23.
25. Show a request to retrieve the file at location /bin/etc/file!. The client needs the document only if it was modified after January 23, 1999.
26. Show the response to Exercise 25.
27. Show a request to retrieve the file at location /bin/etc/filel. The client should identify itself.
28. Show the response to Exercise 27.
29. Show a request to store a file at location /bin/letter. The client identifies the types of documents it can accept.
30. Show the response to Exercise 29. The response shows the age of the document as well as the date and time when the contents may change.

CHAPTER 28

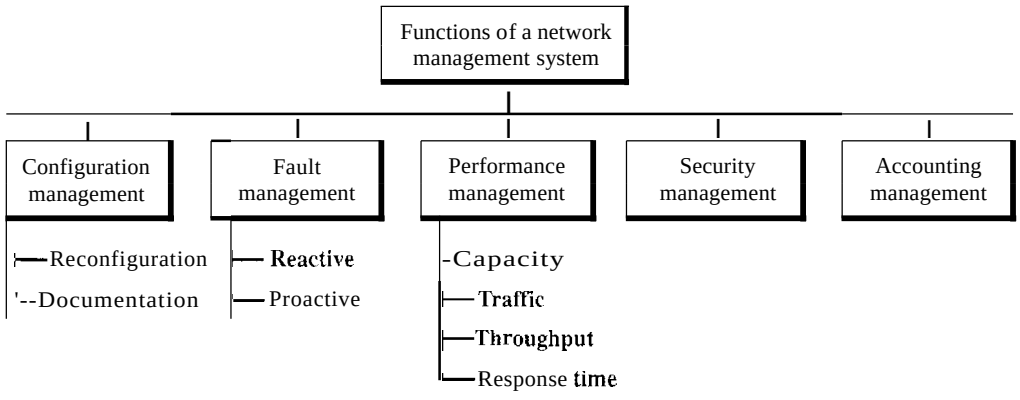
Network Management: SNMP

We can define **network management** as monitoring, testing, configuring, and troubleshooting network components to meet a set of requirements defined by an organization. These requirements include the smooth, efficient operation of the network that provides the predefined quality of service for users. To accomplish this task, a network management system uses hardware, software, and humans. In this chapter, first we briefly discuss the functions of a network management system. Then we concentrate on the most common management system, the Simple Network Management Protocol (SNMP).

28.1 NETWORK MANAGEMENT SYSTEM

We can say that the functions performed by a network management system can be divided into five broad categories: configuration management, fault management, performance management, security management, and accounting management, as shown in Figure 28.1.

Figure 28.1 *Functions of a network management system*



Configuration Management

A large network is usually made up of hundreds of entities that are physically or logically connected to one another. These entities have an initial configuration when the network is set up, but can change with time. Desktop computers may be replaced by others; application software may be updated to a newer version; and users may move from one group to another. The **configuration management** system must know, at any time, the status of each entity and its relation to other entities. Configuration management can be divided into two subsystems: reconfiguration and documentation.

Reconfiguration

Reconfiguration, which means adjusting the network components and features, can be a daily occurrence in a large network. There are three types of reconfiguration: hardware reconfiguration, software reconfiguration, and user-account reconfiguration.

Hardware reconfiguration covers all changes to the hardware. For example, a desktop computer may need to be replaced. A router may need to be moved to another part of the network. A subnetwork may be added or removed from the network. All these need the time and attention of network management. In a large network, there must be specialized personnel trained for quick and efficient hardware reconfiguration. Unfortunately, this type of reconfiguration cannot be automated and must be manually handled case by case.

Software reconfiguration covers all changes to the software. For example, new software may need to be installed on servers or clients. An operating system may need updating. Fortunately, most software reconfiguration can be automated. For example, updating an application on some or all clients can be electronically downloaded from the server.

User-account reconfiguration is not simply adding or deleting users on a system. You must also consider the user privileges, both as an individual and as a member of a group. For example, a user may have read and write permission with regard to some files, but only read permission with regard to other files. User-account reconfiguration can be, to some extent, automated. For example, in a college or university, at the beginning of each quarter or semester, new students are added to the system. The students are normally grouped according to the courses they take or the majors they pursue.

Documentation

The original network configuration and each subsequent change must be recorded meticulously. This means that there must be documentation for hardware, software, and user accounts.

Hardware documentation normally involves two sets of documents: maps and specifications. Maps track each piece of hardware and its connection to the network. There can be one general map that shows the logical relationship between each subnetwork. There can also be a second general map that shows the physical location of each subnetwork. For each subnetwork, then, there is one or more maps that show all pieces of equipment. The maps use some kind of standardization to be easily read and understood by current and future personnel. Maps are not enough per se. Each piece of hardware also needs to be documented. There must be a set of specifications for each piece

of hardware connected to the network. These specifications must include information such as hardware type, serial number, vendor (address and phone number), time of purchase, and warranty information.

All software must also be documented. Software documentation includes information such as the software type, the version, the time installed, and the license agreement.

Most operating systems have a utility that allows the documentation of user accounts and their privileges. The management must make sure that the files with this information are updated and secured. Some operating systems record access privileges in two documents; one shows all files and access types for each user; the other shows the list of users that have a particular access to a file.

Fault Management

Complex networks today are made up of hundreds and sometimes thousands of components. Proper operation of the network depends on the proper operation of each component individually and in relation to each other. **Fault management** is the area of network management that handles this issue.

An effective fault management system has two subsystems: reactive fault management and proactive fault management.

Reactive Fault Management

A reactive fault management system is responsible for detecting, isolating, correcting, and recording faults. It handles short-term solutions to faults.

The first step taken by a reactive fault management system is to detect the exact location of the fault. A fault is defined as an abnormal condition in the system. When a fault occurs, either the system stops working properly or the system creates excessive errors. A good example of a fault is a damaged communication medium. This fault may interrupt communication or produce excessive errors.

The next step taken by a reactive fault management system is to isolate the fault. A fault, if isolated, usually affects only a few users. After isolation, the affected users are immediately notified and given an estimated time of correction.

The third step is to correct the fault. This may involve replacing or repairing the faulty component(s).

After the fault is corrected, it must be documented. The record should show the exact location of the fault, the possible cause, the action or actions taken to correct the fault, the cost, and time it took for each step. Documentation is extremely important for several reasons:

- The problem may recur. Documentation can help the present or future administrator or technician solve a similar problem.
- The frequency of the same kind of failure is an indication of a major problem in the system. If a fault happens frequently in one component, it should be replaced with a similar one, or the whole system should be changed to avoid the use of that type of component.
- The statistic is helpful to another part of network management, performance management.

Proactive Fault Management

Proactive fault management tries to prevent faults from occurring. Although this is not always possible, some types of failures can be predicted and prevented. For example, if a manufacturer specifies a lifetime for a component or a part of a component, it is a good strategy to replace it before that time. As another example, if a fault happens frequently at one particular point of a network, it is wise to carefully reconfigure the network to prevent the fault from happening again.

Performance Management

Performance management, which is closely related to fault management, tries to monitor and control the network to ensure that it is running as efficiently as possible. Performance management tries to quantify performance by using some measurable quantity such as capacity, traffic, throughput, or response time.

Capacity

One factor that must be monitored by a performance management system is the capacity of the network. Every network has a limited capacity, and the performance management system must ensure that it is not used above this capacity. For example, if a LAN is designed for 100 stations at an average data rate of 2 Mbps, it will not operate properly if 200 stations are connected to the network. The data rate will decrease and blocking may occur.

Traffic

Traffic can be measured in two ways: internally and externally. Internal traffic is measured by the number of packets (or bytes) traveling inside the network. External traffic is measured by the exchange of packets (or bytes) outside the network. During peak hours, when the system is heavily used, blocking may occur if there is excessive traffic.

Throughput

We can measure the throughput of an individual device (such as a router) or a part of the network. Performance management monitors the throughput to make sure that it is not reduced to unacceptable levels.

Response Time

Response time is normally measured from the time a user requests a service to the time the service is granted. Other factors such as capacity and traffic can affect the response time. Performance management monitors the average response time and the peak-hour response time. Any increase in response time is a very serious condition as it is an indication that the network is working above its capacity.

Security Management

Security management is responsible for controlling access to the network based on the predefined policy. We discuss security and in particular network security in Chapters 31 and 32.

Accounting Management

Accounting management is the control of users' access to network resources through charges. Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network. Charging does not necessarily mean cash transfer; it may mean debiting the departments or divisions for budgeting purposes. Today, organizations use an accounting management system for the following reasons:

- It prevents users from monopolizing limited network resources.
- It prevents users from using the system inefficiently.
- Network managers can do short- and long-term planning based on the demand for network use.

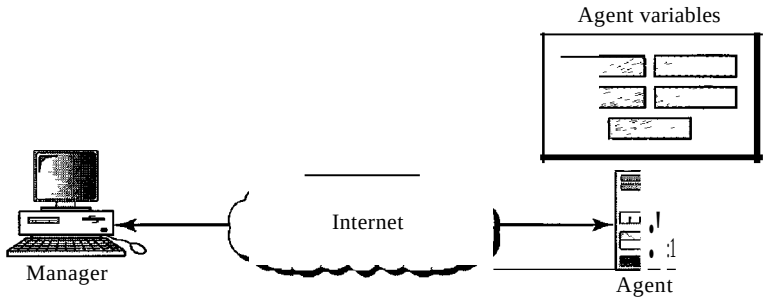
28.2 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

The Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the TCPIIP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet.

Concept

SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers (see Figure 28.2).

Figure 28.2 *SNMP concept*



SNMP is an application-level protocol in which a few manager stations control a set of agents. The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks. In other words, SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology. It can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers.

Managers and Agents

A management station, called a manager, is a host that runs the SNMP client program. A managed station, called an agent, is a router (or a host) that runs the SNMP server program. Management is achieved through simple interaction between a manager and an agent.

The agent keeps performance information in a database. The manager has access to the values in the database. For example, a router can store in appropriate variables the number of packets received and forwarded. The manager can fetch and compare the values of these two variables to see if the router is congested or not.

The manager can also make the router perform certain actions. For example, a router periodically checks the value of a reboot counter to see when it should reboot itself. It reboots itself, for example, if the value of the counter is 0. The manager can use this feature to reboot the agent remotely at any time. It simply sends a packet to force a 0 value in the counter.

Agents can also contribute to the management process. The server program running on the agent can check the environment, and if it notices something unusual, it can send a warning message, called a trap, to the manager.

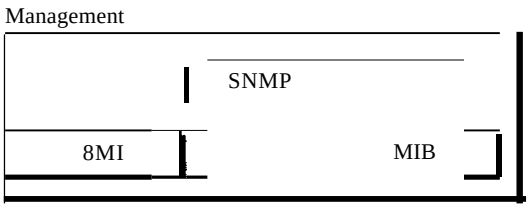
In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by warning the manager of an unusual situation.

Management Components

To do management tasks, SNMP uses two other protocols: Structure of Management Information (SMI) and Management Information Base (MIB). In other words, management on the Internet is done through the cooperation of the three protocols SNMP, SMI, and MIB, as shown in Figure 28.3.

Figure 28.3 *Components of network management on the Internet*



Let us elaborate on the interactions between these protocols.

Role of SNMP

SNMP has some very specific roles in network management. It defines the format of the packet to be sent from a manager to an agent and vice versa. It also interprets the

result and creates statistics (often with the help of other management software). The packets exchanged contain the object (variable) names and their status (values). SNMP is responsible for reading and changing these values.

SNMP defines the format of packets exchanged between a manager and an agent. It reads and changes the status (values) of objects (variables) in 8NMP packets.

Role of SMI

To use SNMP, we need rules. We need rules for naming objects. This is particularly important because the objects in SNMP form a hierarchical structure (an object may have a parent object and some children objects). Part of a name can be inherited from the parent. We also need rules to define the type of the objects. What types of objects are handled by SNMP? Can SNMP handle simple types or structured types? How many simple types are available? What are the sizes of these types? What is the range of these types? In addition, how are each of these types encoded?

We need these universal rules because we do not know the architecture of the computers that send, receive, or store these values. The sender may be a powerful computer in which an integer is stored as 8-byte data; the receiver may be a small computer that stores an integer as 4-byte data.

SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type. SMI is a collection of general rules to name objects and to list their types. The association of an object with the type is not done by SMI.

8MI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.

8M1 does not define the number of objects an entity should manage or name the objects to be managed or define the association between the objects and their values.

Role of MIB

We hope it is clear that we need another protocol. For each entity to be managed, this protocol must define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object. This protocol is MIB. MIB creates a set of objects defined for each entity similar to a database (mostly metadata in a database, names and types without values).

MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed.

An Analogy

Before discussing each of these protocols in greater detail, we give an analogy. The three network management components are similar to what we need when we write a program in a computer language to solve a problem.

Before we write a program, the syntax of the language (such as C or Java) must be predefined. The language also defines the structure of variables (simple, structured, pointer, and so on) and how the variables must be named. For example, a variable name must be 1 to *N* characters in length and start with a letter followed by alphanumeric characters. The language also defines the type of data to be used (integer, float, char, etc.). In programming the rules are defined by the language. In network management the rules are defined by SMI.

Most computer languages require that variables be declared in each specific program. The declaration names each variable and defines the predefined type. For example, if a program has two variables (an integer named *counter* and an array named *grades* of type char), they must be declared at the beginning of the program:

```
int counter;
char grades [40];
```

Note that the declarations name the variables (counter and grades) and define the type of each variable. Because the types are predefined in the language, the program knows the range and size of each variable.

MIB does this task in network management. MIB names each object and defines the type of the objects. Because the type is defined by SMI, SNMP knows the range and size.

After declaration in programming, the program needs to write statements to store values in the variables and change them if needed. SNMP does this task in network management. SNMP stores, changes, and interprets the values of objects already declared by MIB according to the rules defined by SMI.

We can compare the task of network management to the task of writing a program.

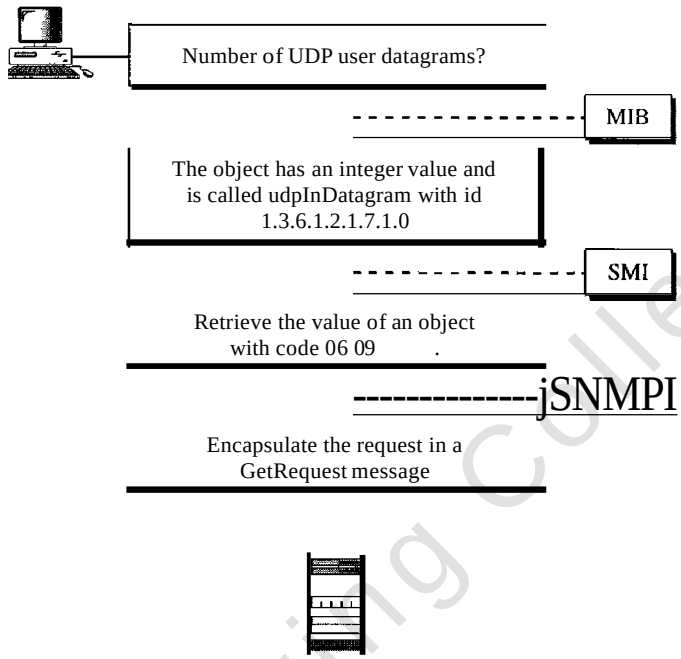
- Both tasks need rules. In network management this is handled by SMI.
- Both tasks need variable declarations. In network management this is handled by MIB.
- Both tasks have actions performed by statements. In network management this is handled by SNMP.

An Overview

Before discussing each component in detail, we show how each is involved in a simple scenario. This is an overview that will be developed later at the end of the chapter. A manager station (SNMP client) wants to send a message to an agent station (SNMP server) to find the number of UDP user datagrams received by the agent. Figure 28.4 shows an overview of steps involved.

MIB is responsible for finding the object that holds the number of the UDP user datagrams received. SMI, with the help of another embedded protocol, is responsible for encoding the name of the object. SNMP is responsible for creating a message, called a GetRequest message, and encapsulating the encoded message. Of course, things are more complicated than this simple overview, but we first need more details of each protocol.

Figure 28.4 *Management overview*



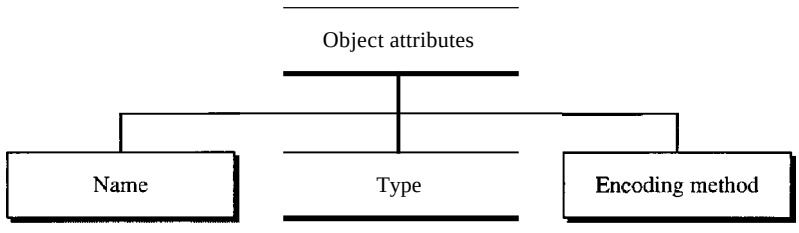
Structure of Management Information

The Structure of Management Information, version 2 (SMIv2) is a component for network management. Its functions are

1. To name objects
2. To define the type of data that can be stored in an object
3. To show how to encode data for transmission over the network

SMI is a guideline for SNMP. It emphasizes three attributes to handle an object: name, data type, and encoding method (see Figure 28.5).

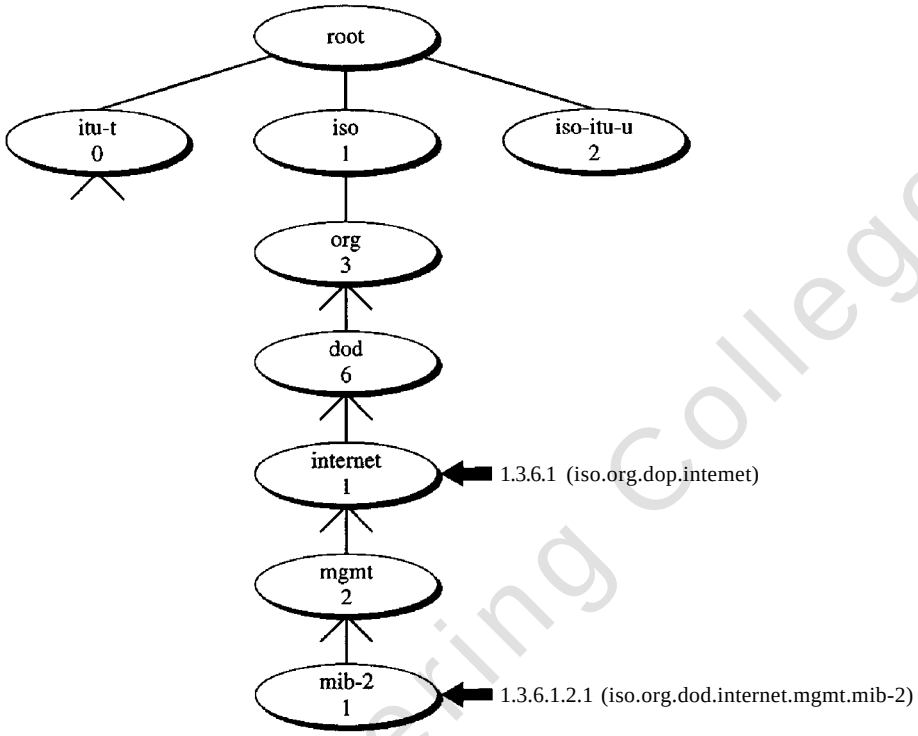
Figure 28.5 *Object attributes*



Name

SMI requires that each managed object (such as a router, a variable in a router, a value) have a unique name. To name objects globally, SMI uses an object identifier, which is a hierarchical identifier based on a tree structure (see Figure 28.6).

Figure 28.6 *Object identifier*



The tree structure starts with an unnamed root. Each object can be defined by using a sequence of integers separated by dots. The tree structure can also define an object by using a sequence of textual names separated by dots. The integer-dot representation is used in SNMP. The name-dot notation is used by people. For example, the following shows the same object in two different notations:

iso.org.dod.internet.mgmt.mib-2 ➡ 1.3.6.1.2.1

The objects that are used in SNMP are located under the *mib-2* object, so their identifiers always start with 1.3.6.1.2.1.

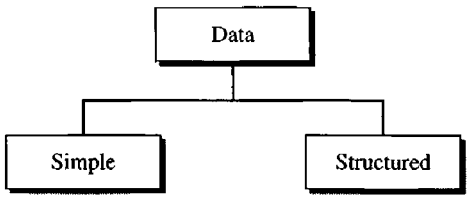
AU objects managed by SNMP are given an object identifier.
The object identifier always starts with 1.3.6.1.2.1.

Type

The second attribute of an object is the type of data stored in it. To define the data type, SMI uses fundamental Abstract Syntax Notation 1 (ASN.l) definitions and adds some new definitions. In other words, SMI is both a subset and a superset of ASN.1.

SMI has two broad categories of data type: *simple* and *structured.* We first define the simple types and then show how the structured types can be constructed from the simple ones (see Figure 28.7).

Figure 28.7 *Data type*



Simple Type The simple data types are atomic data types. Some of them are taken directly from ASN.l; others are added by SMI. The most important ones are given in Table 28.1. The first five are from ASN.l; the next seven are defined by SMI.

Table 28.1 *Data types*

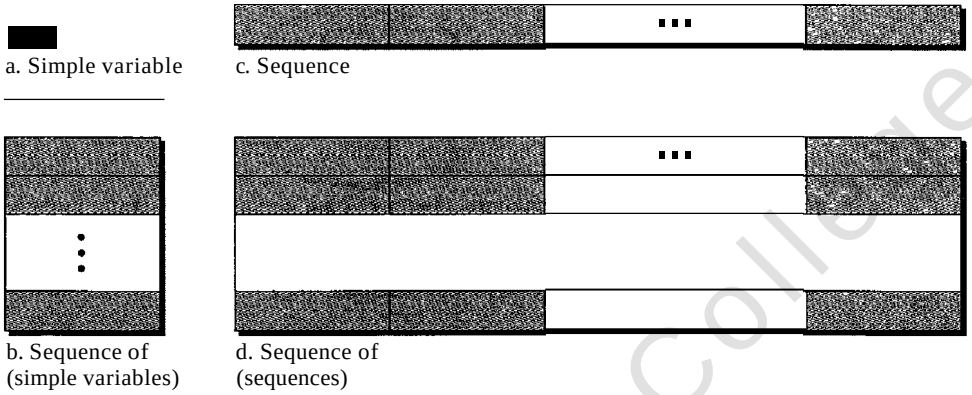
Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31} - 1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32} - 1$
OCTET STRING	Variable	Byte string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}; when it reaches its maximum value, it wraps back to 0.
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in $\frac{1}{100}$ s
BITS		A string of bits
Opaque	Variable	Uninterpreted string

Structured Type By combining simple and structured data types, we can make new structured data types. SMI defines two structured data types: *sequence* and *sequence of*

- Sequence. A *sequence* data type is a combination of simple data types, not necessarily of the same type. It is analogous to the concept of a *struct* or a *record* used in programming languages such as C.
- Sequence of. A *sequence of* data type is a combination of simple data types all of the same type or a combination of sequence data types all of the same type. It is analogous to the concept of an *array* used in programming languages such as C.

Figure 28.8 shows a conceptual view of data types.

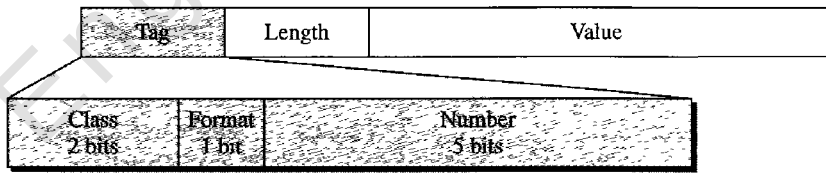
Figure 28.8 *Conceptual data types*



Encoding Method

SMI uses another standard, Basic Encoding Rules (BER), to encode data to be transmitted over the network. BER specifies that each piece of data be encoded in triplet format: tag, length, and value, as illustrated in Figure 28.9.

Figure 28.9 *Encoding format*



- Tag. The tag is a I-byte field that defines the type of data. It is composed of three subfields: *class* (2 *bits),fonnat* (1 bit), and *number* (5 bits). The class subfield defines the scope of the data. Four classes are defined: universal (00), applicationwide (01), context-specific (10), and private (11). The universal data types are those taken from ASN.l (INTEGER, OCTET STRING, and ObjectIdentifier). The applicationwide data types are those added by SMI (IPAddress, Counter, Gauge, and TimeTicks). The five context-specific data types have meanings that may change from one protocol to another. The private data types are vendor-specific.

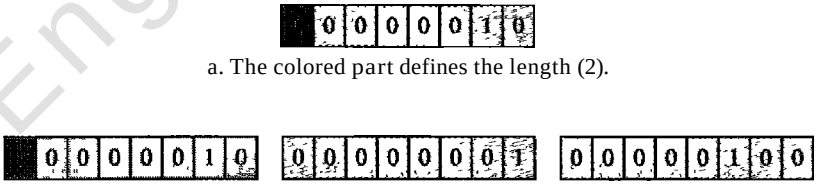
  The format subfield indicates whether the data are simple (0) or structured (1). The number subfield further divides simple or structured data into subgroups. For example, in the universal class, with simple format, INTEGER has a value of 2, OCTET STRING has a value of 4, and so on. Table 28.2 shows the data types we use in this chapter and their tags in binary and hexadecimal numbers.

Table 28.2 *Codes for data types*

Data Type	*Class*	*Format*	*Number*	*Tag (Binary)*	*Tag (Hex)*
INTEGER	00	0	00010	00000010	02
OCTET STRING	00	0	00100	00000100	04
OBJECT IDENTIFIER	00	0	00110	00000110	06
NULL	00	0	00101	00000101	05
Sequence, sequence of	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43
Opaque	01	0	00100	01000100	44

- Length. The length field is 1 or more bytes. If it is 1 byte, the most significant bit must be 0. The other 7 bits define the length of the data. If it is more than 1 byte, the most significant bit of the first byte must be 1. The other 7 bits of the first byte define the number of bytes needed to define the length. See Figure 28.10 for a depiction of the length field.

Figure 28.10 *Length format*



a. The colored part defines the length (2).

b. The shaded part dermes the length of the length (2 bytes); the colored bytes define the length (260 bytes).

- Value. The value field codes the value of the data according to the rules defined in BER.

To show how these three fields-tag, length, and value-can define objects, we give some examples.

Example 28.1

Figure 28.11 shows how to define INTEGER 14.

Example 28.2

Figure 28.12 shows how to define the OCTET STRING "HI."

Figure 28.11 *Example* 28.1, *INTEGER 14*

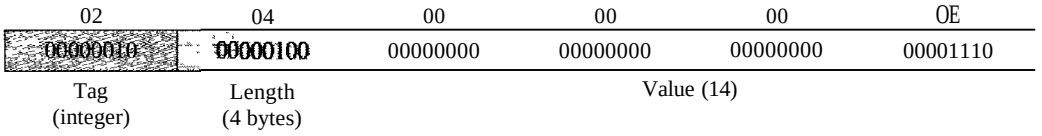
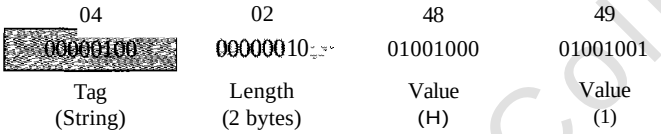


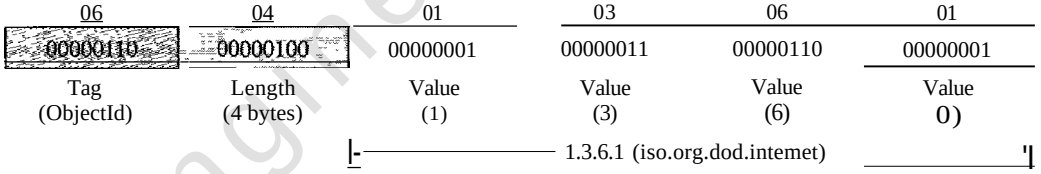
Figure 28.12 *Example* 28.2, *OCTET STRING "HI"*



Example 28.3

Figure 28.13 shows how to define ObjectIdentifier 1.3.6.1 (iso.org.dod.internet).

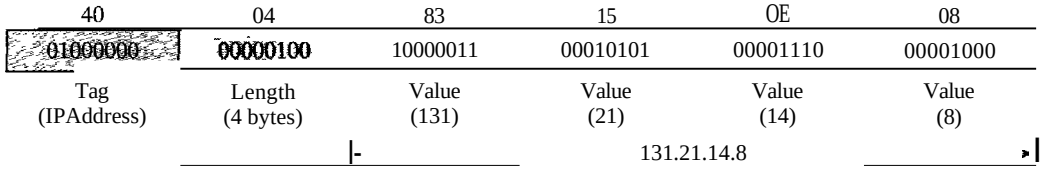
Figure 28.13 *Example* 28.3, *ObjectIdentifier 1.3.6.1*



Example 28.4

Figure 28.14 shows how to define IPAddress 131.21.14.8.

Figure 28.14 *Example* 28.4, *IPAddress 131.21.14.8*

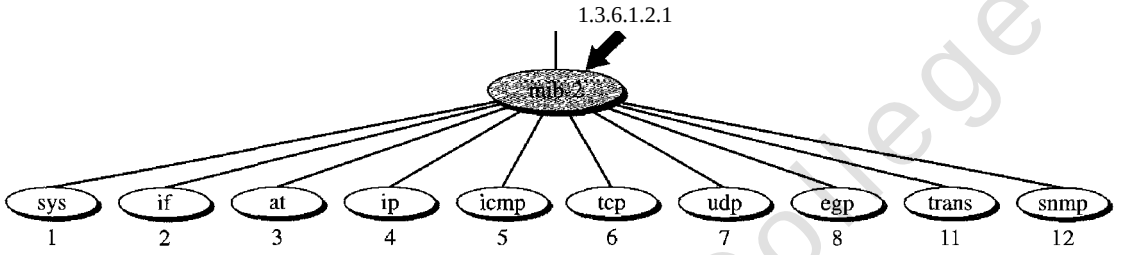


Management Information Base (MIB)

The Management Information Base, version 2 (MIB2) is the second component used in network management. Each agent has its own MIB2, which is a collection of all the objects that the manager can manage. The objects in MIB2 are categorized under 10

different groups: system, interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp. These groups are under the mib-2 object in the object identifier tree (see Figure 28.15). Each group has defined variables and/or tables.

Figure 28.15 *mib-2*



The following is a brief description of some of the objects:

- sys This object *(system)* defines general information about the node (system), such as the name, location, and lifetime.
- if This object *(inteiface)* defines information about all the interfaces of the node including interface number, physical address, and IP address.
- at This object *(address translation)* defines the information about the ARP table.
- ip This object defines information related to IP, such as the routing table and the IP address.
- icmp This object defines information related to ICMP, such as the number of packets sent and received and total errors created.
- tcp This object defines general information related to TCP, such as the connection table, time-out value, number of ports, and number of packets sent and received.
- udp This object defines general information related to UDP, such as the number of ports and number of packets sent and received.
- snmp This object defines general information related to SNMP itself.

Accessing MIB Variables

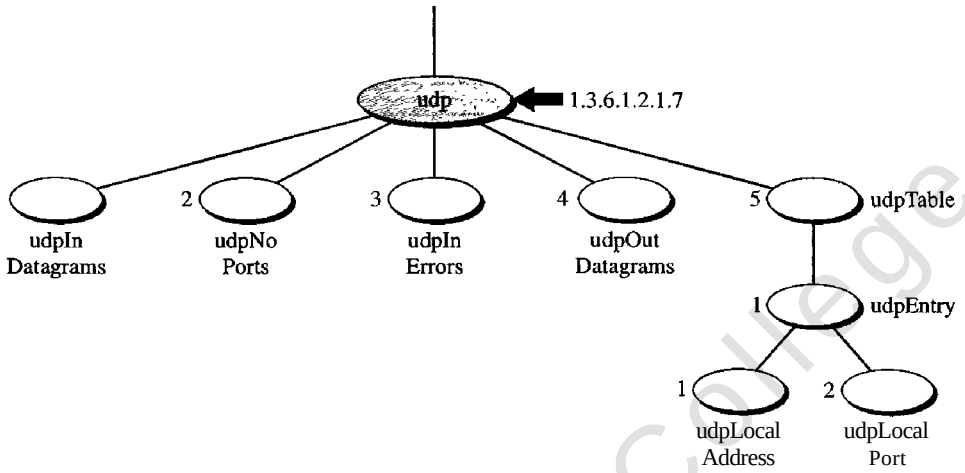
To show how to access different variables, we use the udp group as an example. There are four simple variables in the udp group and one sequence of (table of) records. Figure 28.16 shows the variables and the table.

We will show how to access each entity.

Simple Variables To access any of the simple variables, we use the id of the group (1.3.6.1.2.1.7) followed by the id of the variable. The following shows how to access each variable.

udpInDatagrams → 1.3,6.1.2.1.7.1
udpNoPorts → 1.3.6.1.2.1.7.2
udplnErrors → 1.3.6.1.2.1.7.3
udpOutDatagrams → 1.3.6.1.2.1.7.4

Figure 28.16 *udp group*



However, these object identifiers define the variable, not the instance (contents). To show the instance or the contents of each variable, we must add an instance suffix. The instance suffix for a simple variable is simply a 0. In other words, to show an instance of the above variables, we use the following:

udpInDatagrams.O	1.3.6.1.2.1.7.1.0
udpNoPorts.O	1.3.6.1.2.1.7.2.0
udpInErrors.0	1.3.6.1.2.1.7.3.0
udpOutDatagrams.0	1.3.6.1.2.1.7.4.0

Tables To identify a table, we first use the table id. The udp group has only one table (with id 5) as illustrated in Figure 28.17.

So to access the table, we use the following:

udpTable ➡ 1.3.6.1.2.1.7.5

However, the table is not at the leaf level in the tree structure. We cannot access the table; we define the entry (sequence) in the table (with id of 1), as follows:

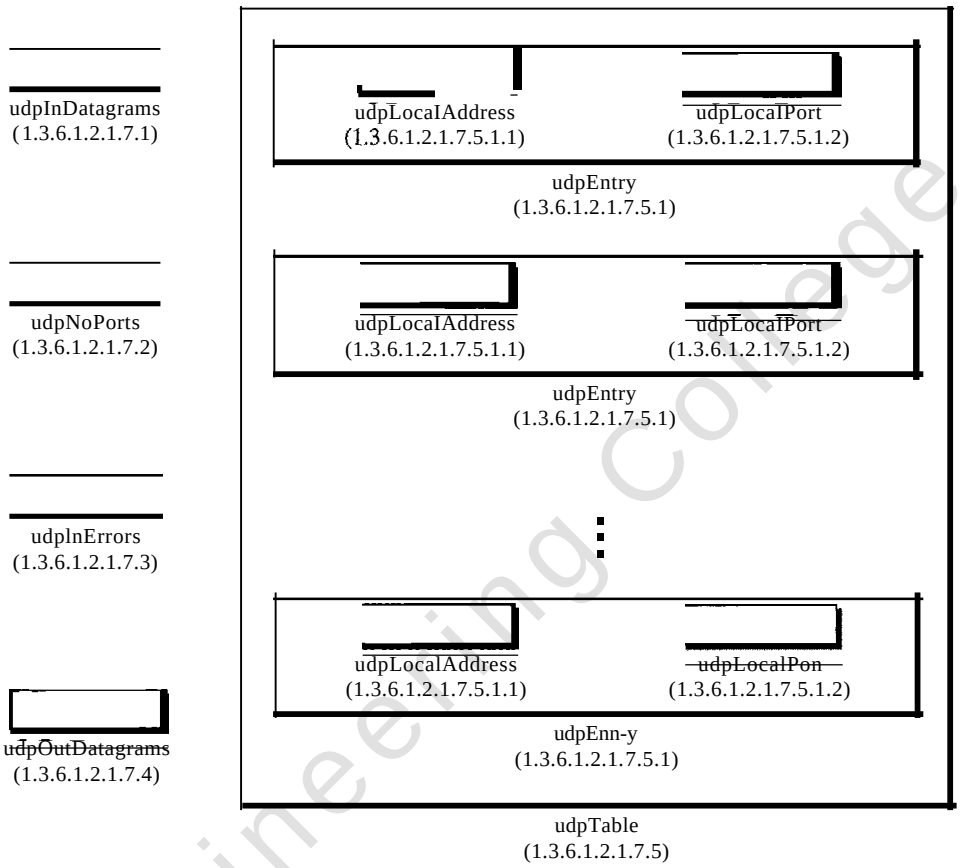
udpEntry ➡ 1.3.6.1.2.1.7.5.1

This entry is also not a leaf and we cannot access it. We need to define each entity (field) in the entry.

udpLocalAddress	1.3.6.1.2.1.7.5.1.1
udpLocalPort	1.3.6.1.2.1.7.5.1.2

These two variables are at the leaf of the tree. Although we can access their instances, we need to define *which* instance. At any moment, the table can have several values for

Figure 28.17 *udp variables and tables*



each local address/local port pair. To access a specific instance (row) of the table, we add the index to the above ids. In MIB, the indexes of arrays are not integers (like most programming languages). The indexes are based on the value of one or more fields in the entries. In our example, the udpTable is indexed based on both the local address and the local port number. For example, Figure 28.18 shows a table with four rows and values for each field. The index of each row is a combination of two values.

To access the instance of the local address for the first row, we use the identifier augmented with the instance index:

udpLocalAddress.181.23.45.14.23 ➡ 1.3.6.1.2.7.5.1.1.181.23.45.14.23

Note that not all tables are indexed in the same way. Some tables are indexed by using the value of one field, others by using the value of two fields, and so on.

Lexicographic Ordering

One interesting point about the MIB variables is that the object identifiers (including the instance identifiers) follow in lexicographic order. Tables are ordered according to column-row rules, which means one should go column by column. In each column, one should go from the top to the bottom, as shown in Figure 28.19.

Figure 28.18 *Indexes for udpTable*

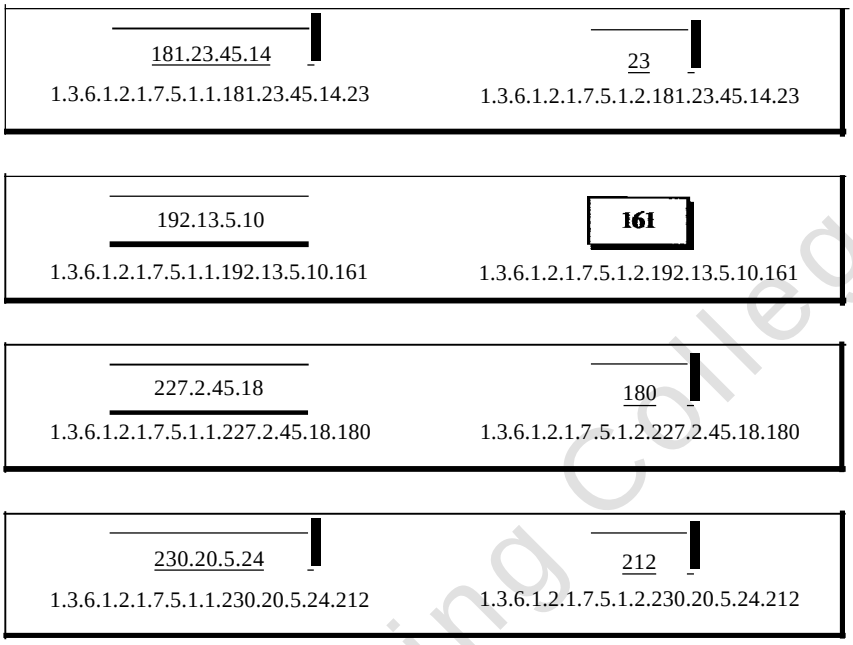
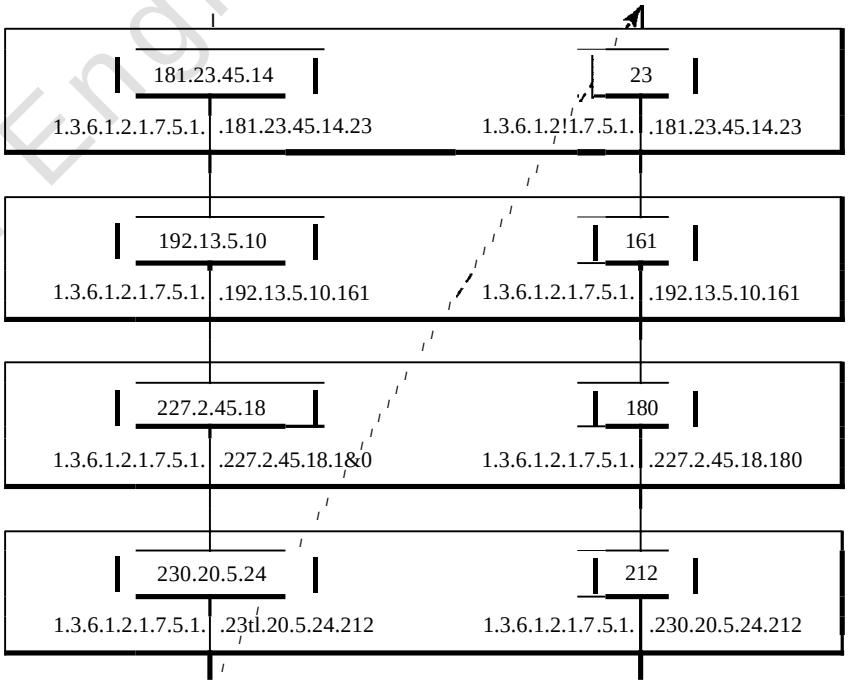


Figure 28.19 *Lexicographic ordering*



The lexicographic ordering enables a manager to access a set of variables one after another by defining the first variable, as we will see in the GetNextRequest command in the next section.

SNMP

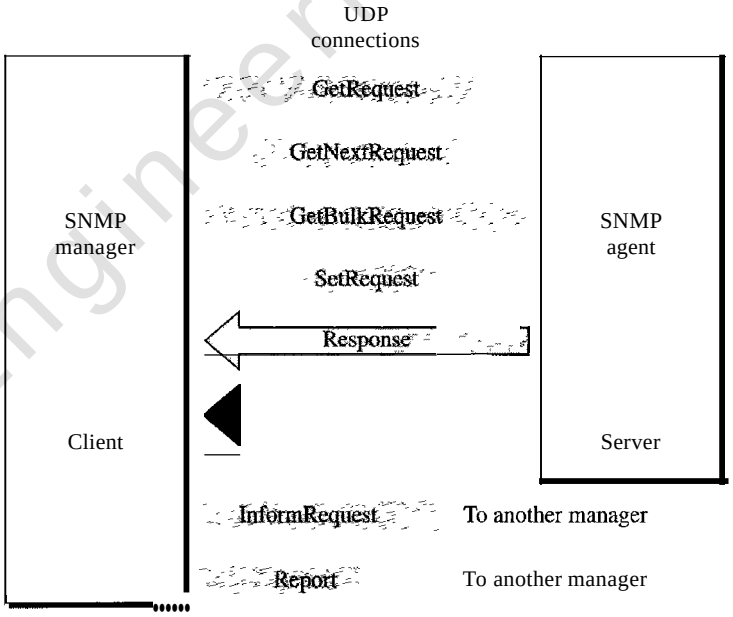
SNMP uses both SMI and MIB in Internet network management. It is an application program that allows

1. A manager to retrieve the value of an object defined in an agent
2. A manager to store a value in an object defined in an agent
3. An agent to send an alarm message about an abnormal situation to the manager

PDUs

SNMPv3 defines eight types of packets (or PDUs): GetRequest, GetNextRequest, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report (see Figure 28.20).

Figure 28.20 *SNMP PDUs*



GetRequest The GetRequest PDU is sent from the manager (client) to the agent (server) to retrieve the value of a variable or a set of variables.

GetNextRequest The GetNextRequest PDU is sent from the manager to the agent to retrieve the value of a variable. The retrieved value is the value of the object following the defined Objectid in the PDD. It is mostly used to retrieve the values of the entries in a table. If the manager does not know the indexes of the entries, it cannot retrieve the values. However, it can use GetNextRequest and define the ObjectId of the table. Because the first entry has the ObjectId immediately after the ObjectId of the table, the value of the first entry is returned. The manager can use this ObjectId to get the value of the next one, and so on.

GetBulkRequest The GetBulkRequest POD is sent from the manager to the agent to retrieve a large amount of data. It can be used instead of multiple GetRequest and GetNextRequest PODs.

SetRequest The SetRequest PDD is sent from the manager to the agent to set (store) a value in a variable.

Response The Response PDD is sent from an agent to a manager in response to GetRequest or GetNextRequest. It contains the value(s) of the variable(s) requested by the manager.

Trap The Trap (also called SNMPv2 Trap to distinguish it from SNMPv1 Trap) POD is sent from the agent to the manager to report an event. For example, if the agent is rebooted, it infonns the manager and reports the time of rebooting.

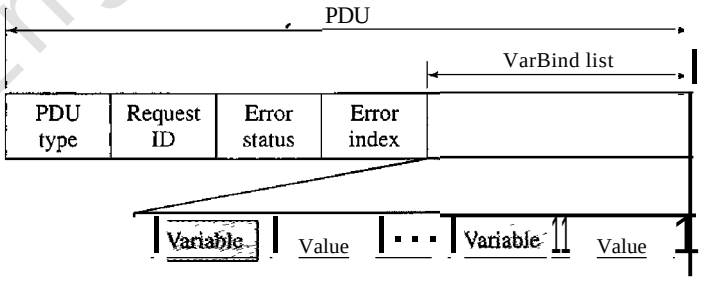
InformRequest The InfonnRequest POD is sent from one manager to another remote manager to get the value of some variables from agents under the control of the remote manager. The remote manager responds with a Response POD.

Report The Report POD is designed to report some types of errors between managers. It is not yet in use.

Format

The fonnat for the eight SNMP PODs is shown in Figure 28.21. The GetBulkRequest POD differs from the others in two areas, as shown in the figure.

Figure 28.21 *SNMP PDU format*



The fields are listed below:

- **PDU** type. This field defines the type of the POD (see Table 28.4).
- Request ID. This field is a sequence number used by the manager in a Request POD and repeated by the agent in a response. It is used to match a request to a response.

- Error status. This is an integer that is used only in Response PDUs to show the types of errors reported by the agent. Its value is 0 in Request PDUs. Table 28.3 lists the types of errors that can occur.

Table 28.3 *Types of errors*

Status	*Name*	*Meaning*
0	noError	No error
1	tooBig	Response too big to fit in one message
2	noSuchName	Variable does not exist
3	badValue	The value to be stored is invalid
4	readOnly	The value cannot be modified
5	genErr	Other errors

- Nonrepeaters. This field is used only in GetBulkRequest and replaces the error status field, which is empty in Request PDUs.
- Error index. The error index is an offset that tells the manager which variable caused the error.
- **Max-repetition.** This field is also used only in GetBulkRequest and replaces the error index field, which is empty in Request PDUs.
- VarBind list. This is a set of variables with the corresponding values the manager wants to retrieve or set. The values are null in GetRequest and GetNextRequest. In a Trap PDU, it shows the variables and values related to a specific PDU.

Messages

SNMP does not send only a PDU, it embeds the PDU in a message. A message in SNMPv3 is made of four elements: version, header, security parameters, and data (which include the encoded PDU), as shown in Figure 28.22.

Because the length of these elements is different from message to message, SNMP uses BER to encode each element. Remember that BER uses the tag and the length to define a value. The *version* defines the current version (3). The *header* contains values for message identification, maximum message size (the maximum size of the reply), message flag (one octet of data type OCTET STRING where each bit defines security type, such as privacy or authentication, or other information), and a message security model (defining the security protocol). The message *security parameter* is used to create a message digest (see Chapter 31). The data contain the PDU. If the data are encrypted, there is information about the encrypting engine (the manager program that did the encryption) and the encrypting context (the type of encryption) followed by the encrypted PDU. If the data are not encrypted, the data consist of just the PDU.

To define the type of PDU, SNMP uses a tag. The class is context-sensitive (10), the format is structured (1), and the numbers are 0, 1, 2, 3, 5, 6, 7, and 8 (see Table 28.4). Note that SNMPv1 defined A4 for Trap, which is obsolete today.

Figure 28.22 *SNMP message*

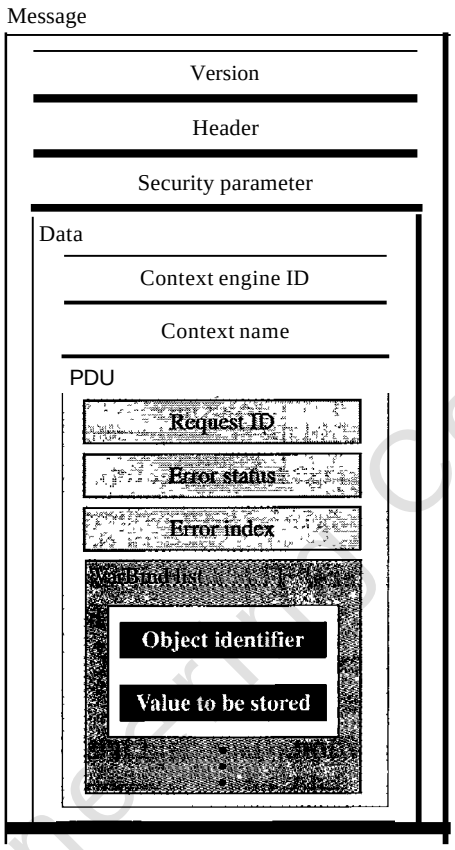


Table 28.4 *Codes for SNMP messages*

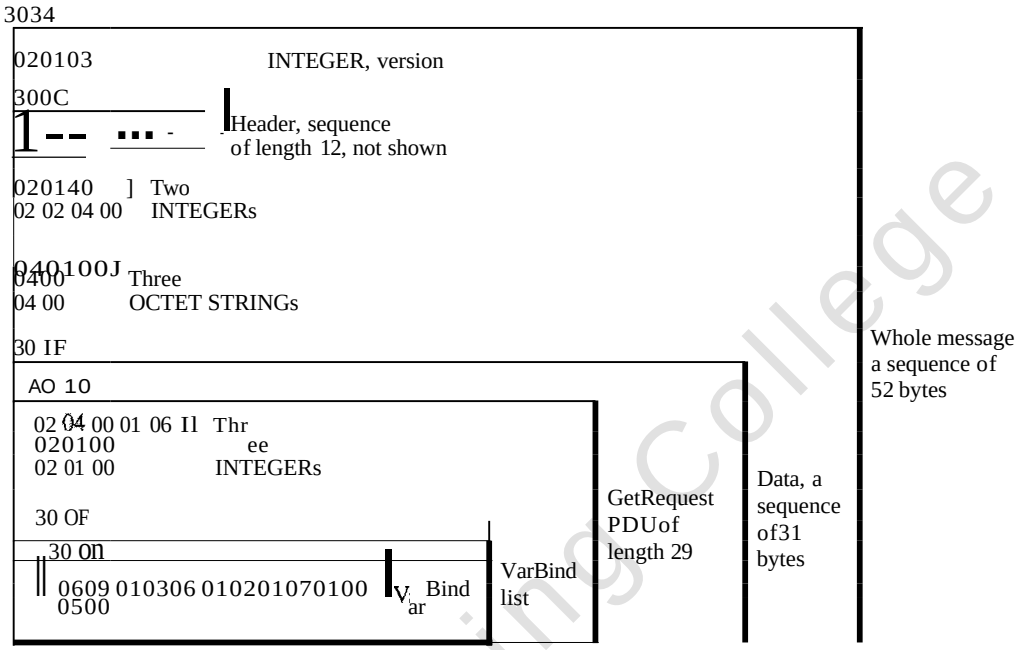
Data	*Class*	*Format*	*Number*	*Whole Tag (Binary)*	*Whole Tag (Hex)*
GetRequest	10	1	00000	**10100000**	**AO**
GetNextRequest	10	1	00001	**10100001**	**Al**
Response	10	1	00010	**10100010**	**A2**
SetRequest	10	1	00011	**10100011**	**A3**
GetBulkRequest	10	1	00101	**10100101**	**AS**
InformRequest	10	1	00110	**10100110**	**A6**
Trap (SNMPv2)	10	1	00111	**10100111**	**A7**
Report	10	1	01000	**10101000**	**A8**

***Example** 28,5*

In this example, a manager station (SNMP client) uses the GetRequest message to retrieve the number of UDP datagrams that a router has received.

There is only one VarBind entity. The corresponding MID variable related to this information is udpInDatagrams with the object identifier 1.3.6.1.2.1.7.1.0. The manager wants to retrieve a value (not to store a value), so the value defines a null entity. Figure 28.23 shows the conceptual

Figure 28.23 *Example 28.5*



view of the packet and the hierarchical nature of sequences. We have used white and colored boxes for the sequences and a gray one for the PDU.

The VarBind list has only one VarBind. The variable is of type 06 and length 09. The value is of type 05 and length 00. The whole VarBind is a sequence of length OD (13). The VarBind list is also a sequence of length OF (15). The GetRequest PDU is of length ID (29).

Now we have three OCTET STRINGs related to the security parameter, security model, and flags. Then we have two integers defining maximum size (1024) and message ID (64). The header is a sequence of length 12, which we left blank for simplicity. There is one integer, version (version 3). The whole message is a sequence of 52 bytes.

Figure 28.24 shows the actual message sent by the manager station (client) to the agent (server).

UDPPorts

SNMP uses the services of UDP on two well-known ports, 161 and 162. The well-known port 161 is used by the server (agent), and the well-known port 162 is used by the client (manager).

The agent (server) issues a passive open on port 161. It then waits for a connection from a manager (client). A manager (client) issues an active open, using an ephemeral port, The request messages are sent from the client to the server, using the ephemeral port as the source port and the well-known port 161 as the destination port. The response messages are sent from the server to the client, using the well-known port 161 as the source port and the ephemeral port as the destination port.

The manager (client) issues a passive open on port 162. It then waits for a connection from an agent (server). Whenever it has a Trap message to send, an agent (server) issues an active open, using an ephemeral port. This connection is only one-way, from the server to the client (see Figure 28.25).

Figure 28.24 *GetRequest message*

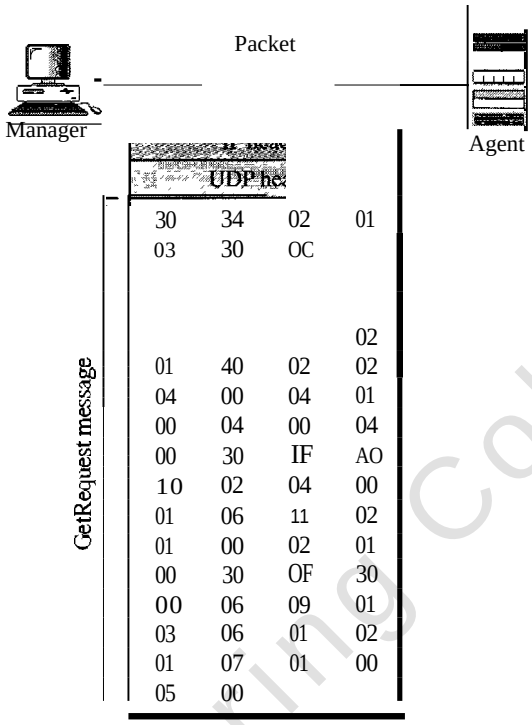
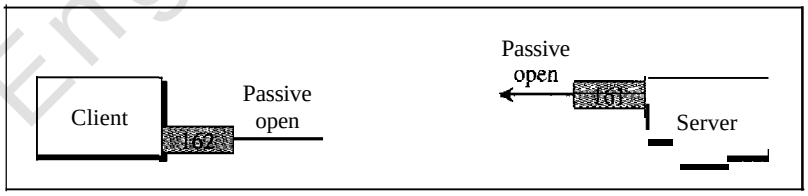
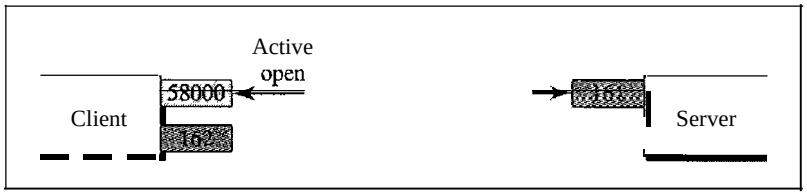


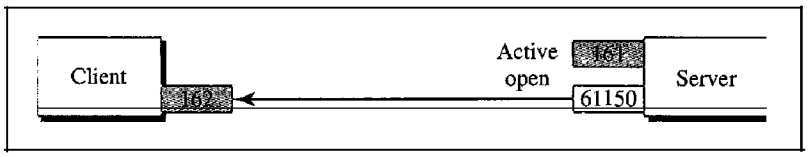
Figure 28.25 *Port numbers for SNMP*



a. Passive open by both client and server



b. Exchange of request and response messages



c. Server sends trap message

The client/server mechanism in SNMP is different from other protocols. Here both the client and the server use well-known ports. In addition, both the client and the server are running infinitely. The reason is that request messages are initiated by a manager (client), but Trap messages are initiated by an agent (server).

Security

The main difference between SNMPv3 and SNMPv2 is the enhanced security. SNMPv3 provides two types of security: general and specific. SNMPv3 provides message authentication, privacy, and manager authorization. We discuss these three aspects in Chapter 31. In addition, SNMPv3 allows a manager to remotely change the security configuration, which means that the manager does not have to be physically present at the manager station.

28.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

SNMP is discussed in [MS01], Chapter 25 of [Ste94], Section 22.3 of [Sta04], and Chapter 39 of [Com04]. Network management is discussed in [Sub01].

Sites

The following sites are related to topics discussed in this chapter.

- www.ietf.org/rfc.html Information about RFCs

RFCs

The following RFCs are related to SNMP, MIB, and SMI:

1065, 1067,1098, 1155,1157, 1212, 1213, 1229, 1231, 1243, 1284, 1351, 1352,1354, 1389, 1398, 1414, 1441, 1442, 1443, 1444,1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1461, 1472, 1474, 1537, 1623,1643,1650, 1657, 1665, 1666, 1696, 1697, 1724, 1742,1743, 1748, 1749

28.4 KEY TERMS

Abstract Syntax Notation 1 (ASN.l)
accounting management
agent
Basic Encoding Rules (BER)
configuration management
fault management
hardware documentation
lexicographic ordering
Management Information Base (MIB)
manager
network management
object identifier
performance management
security management

simple data type
Simple Network Management Protocol (SNMP)
structured data type
Structure of Management Information (SMI)
trap

28.5 SUMMARY

- The five areas comprising network management are configuration management, fault management, performance management, accounting management, and security management.
- Configuration management is concerned with the physical or logical changes of network entities. It includes the reconfiguration and documentation of hardware, software, and user accounts.
- Fault management is concerned with the proper operation of each network component. It can be reactive or proactive.
- Performance management is concerned with the monitoring and control of the network to ensure the network runs as efficiently as possible. It is quantified by measuring the capacity, traffic, throughput, and response time.
- Security management is concerned with controlling access to the network.
- Accounting management is concerned with the control of user access to network resources through charges.
- Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the TCP/IP protocol suite.
- A manager, usually a host, controls and monitors a set of agents, usually routers.
- The manager is a host that runs the SNMP client program.
- The agent is a router or host that runs the SNMP server program.
- SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology.
- SNMP uses the services of two other protocols: Structure of Management Information (SMI) and Management Information Base (MIB).
- SMI names objects, defines the type of data that can be stored in an object, and encodes the data.
- SMI objects are named according to a hierarchical tree structure.
- SMI data types are defined according to Abstract Syntax Notation 1 (ASN.l).
- SMI uses Basic Encoding Rules (BER) to encode data.
- MIB is a collection of groups of objects that can be managed by SNMP.
- MIB uses lexicographic ordering to manage its variables.
- SNMP functions in three ways:
  1. A manager can retrieve the value of an object defined in an agent.
  2. A manager can store a value in an object defined in an agent.
  3. An agent can send an alarm message to the manager.

- SNMP defines eight types of packets: GetRequest, GetNextRequest, SetRequest, GetBulkRequest, Trap, InformRequest, Response, and Report.
- SNMP uses the services of UDP on two well-known ports, 161 and 162.
- SNMPv3 has enhanced security features over previous versions.

28.6 PRACTICE SET

Review Questions

1. Define network management.
2. List five functions of network management.
3. Define configuration management and its purpose.
4. List two subfunctions of configuration management.
5. Define fault management and its purpose.
6. List two subfunctions of fault management.
7. Define performance management and its purpose.
8. List four measurable quantities of performance management.
9. Define security management and its purpose.
10. Define account management and its purpose.

Exercises

11. Show the encoding for INTEGER 1456.
12. Show the encoding for the OCTET STRING "Hello World."
13. Show the encoding for an arbitrary OCTET STRING of length 1000.
14. Show how the following record (sequence) is encoded.

INTEGER	OCTET STRING	IP Address
2345	"COMPUTER"	185.32.1.5

15. Show how the following record (sequence) is encoded.

Time Tick	INTEGER	Object Id
12000	14564	1.3.6.1.2.1.7

16. Show how the following array (sequence of) is encoded. Each element is an integer.

2345
1236
122
1236

17. Show how the following array of records (sequence of sequence) is encoded.

INTEGER	OCTET STRING	Counter
2345	"COMPUTER"	345
1123	"DISK"	1430
3456	"MONITOR"	2313

18. Decode the following.
    a. 02 04 01 02 14 32
    b. 30060201 11 0201 14
    c. 300904034143420202 14 14
    d. 30 OA 40 04 2351 6271 0202 14 12