Arunai Engineering College
Tiruvannamalai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS8591– COMPUTER NETWORKS
PART B - 13 MARKS

CS8591 COMPUTER NETWORKS

UNIT I INTRODUCTION AND PHYSICAL LAYER

Networks – Network Types – Protocol Layering – TCP/IP Protocol suite – OSI Model – Physical Layer: Performance – Transmission media – Switching – Circuit-switched Networks – Packet Switching.

UNIT II DATA-LINK LAYER & MEDIA ACCESS

Introduction – Link-Layer Addressing – DLC Services – Data-Link Layer Protocols – HDLC – PPP - Media Access Control - Wired LANs: Ethernet - Wireless LANs – Introduction – IEEE 802.11, Bluetooth – Connecting Devices.

UNIT III NETWORK LAYER

Network Layer Services – Packet switching – Performance – IPV4 Addresses – Forwarding of IP Packets - Network Layer Protocols: IP, ICMP v4 – Unicast Routing Algorithms – Protocols – Multicasting Basics – IPV6 Addressing – IPV6 Protocol.

UNIT IV TRANSPORT LAYER

Introduction – Transport Layer Protocols – Services – Port Numbers – User Datagram Protocol – Transmission Control Protocol – SCTP.

UNIT V APPLICATION LAYER

WWW and HTTP – FTP – Email –Telnet –SSH – DNS – SNMP.

TOTAL : 45 PERIODS

UNIT 1

FUNDAMENTALS AND LINK LAYER

PART- B

1. Discuss in detail about Internet Architecture. [APR/MAY-2015,17]

(or)

Draw the OSI Network architecture and explain the functionalities of every layer in detail. [NOV/DEC – 2012]

THE 7-LAYER MODEL – OSI network architecture

OSI refers to **Open System Interconnection** that covers all aspects of network communication. It is a standard of ISO. The open system interconnection model is a layered framework. It has seven separate but interrelated layers. Each layer is having unique responsibilities.

Starting at the bottom and working up,

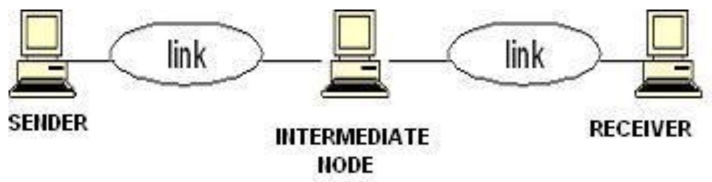
- **The physical layer** handles the transmission of raw bits over a communications link.

It deals with the mechanical and electrical specifications of the interface and transmission medium .

Data rate : no.of bits sent each second – transmission rate is defined by the physical layer

Synchronization of bits : sender and receiver clocks must be synchronized.

- **The data link layer** then collects a stream of bits into a larger aggregate called a frame. Network adaptors, along with device drivers running in the node's operating system, typically implement the data link level.
- **The network layer** handles routing among nodes within a packet-switched network. At this layer, the unit of data exchanged among nodes is typically called a packet rather than a frame,. The lower three layers are implemented on all network nodes, including switches within the network and hosts connected to the exterior of the network.
- **The transport layer** then implements what we have up to this point been calling a process-to-process channel. Here, the unit of data exchanged is commonly called a message rather than a packet or a frame. The transport layer and higher layers typically run only on the end hosts and not on the intermediate switches or routers.



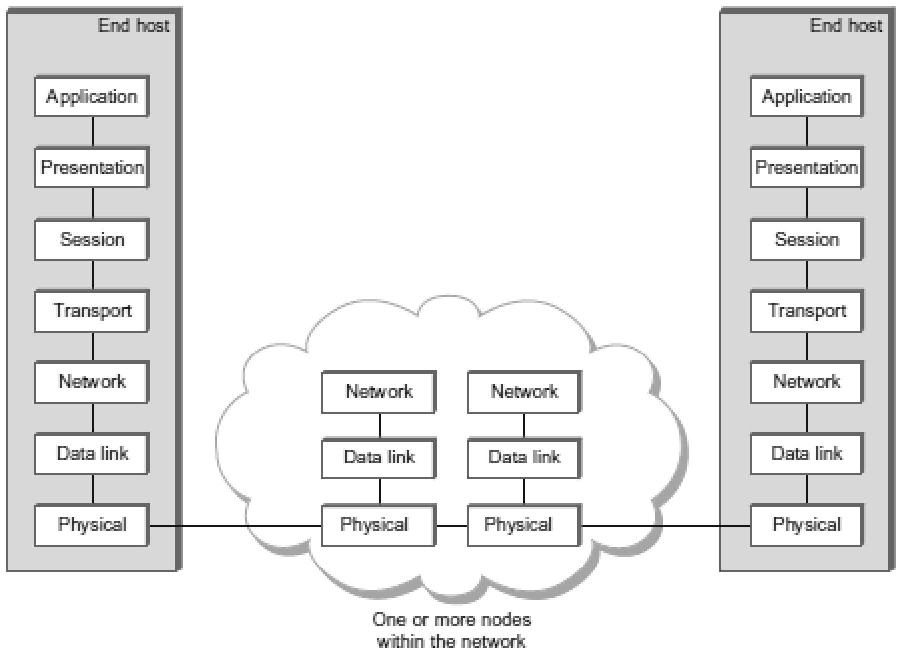


Fig : The OSI 7-layer model

- **The Application layer** protocols include things like the Hypertext Transfer Protocol (HTTP), which is the basis of the World Wide Web and is what enables web browsers to request pages from web servers.
- T**he presentation layer** is concerned with the format of data exchanged between peers—for example, whether an integer is 16, 32, or 64 bits long, whether the most significant byte is transmitted first or last, or how a video stream is formatted.
- **The session layer** provides a name space that is used to tie together the potentially different transport streams that are part of a single application. For example, it might manage an audio stream and a video stream that are being combined in a teleconferencing application.

INTERNET ARCHITECTURE

Internet architecture, which is also sometimes called the TCP/IP architecture after its two main protocols. The internet architecture evolved out of experiences with an earlier packet switched network called the ARPANET. Both the Internet and the ARPANET were funded by the Advanced Research Projects Agency (ARPA).

The Internet and ARPANET were around before the OSI architecture, and the experience gained from building them was a major influence on the OSI reference model. Instead of having seven layers, a four layer model is often used in Internet

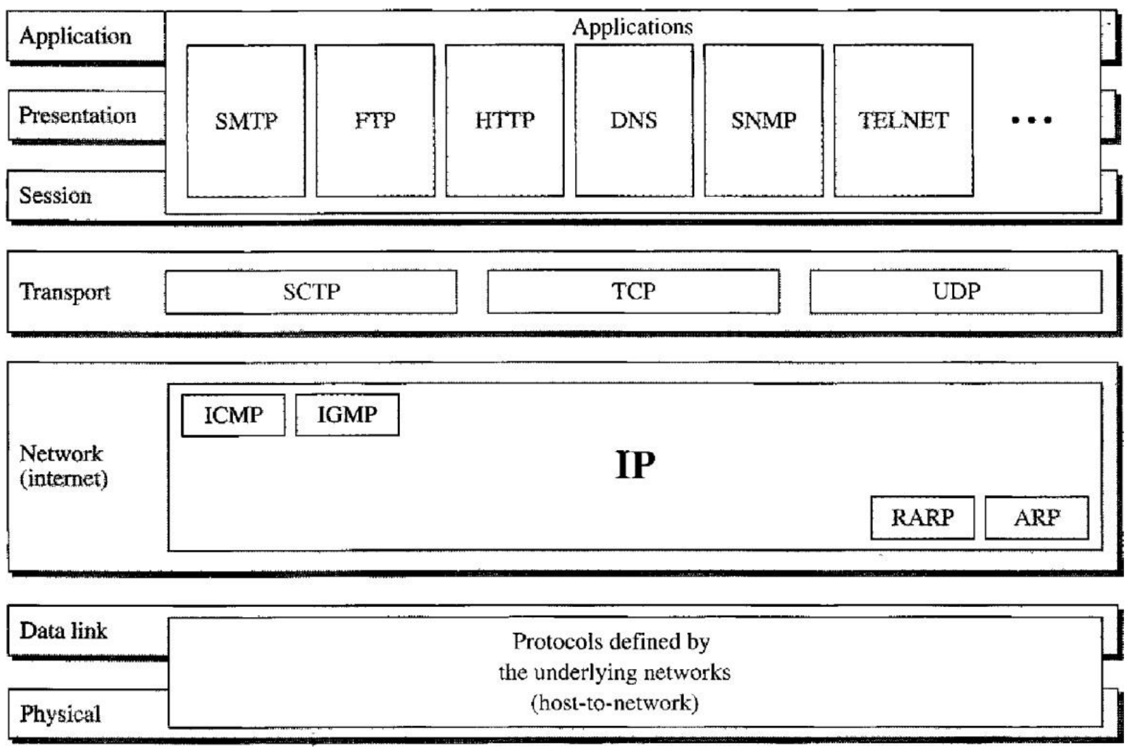


Fig: Internet Architecture

At the lowest level are a wide variety of network protocols, denoted NET1, NET2 and so on. The second layer consists of a single protocol the Internet Protocol IP. It supports the interconnection of multiple networking technologies into a single, logical internetwork.

The third layer contains two main protocols the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP provides a reliable byte stream channel, and UDP provides unreliable datagram delivery channel. They are called as end to end protocol they can also be referred as transport protocols.

Running above the transport layer, a range of application protocols such as FTP, TFTP, Telnet, and SMTP that enable the interoperation of popular applications.

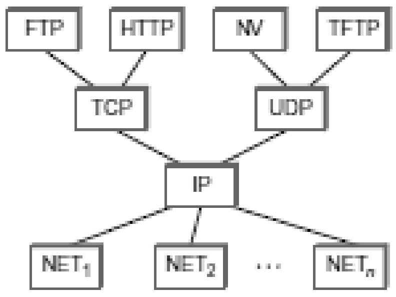


Fig: Internet protocol graph.

The Internet architecture has three features that are worth highlighting.

First, the Internet architecture does not imply strict layering. The application is free to bypass the defined transport layers and to directly use IP or one of the underlying networks. In fact, programmers are free to define new channel abstractions or applications that run on top of any of the existing protocols.

IP serves as the focal point for the architecture— it defines a common method for exchanging packets among a wide collection of networks. Above IP there can be arbitrarily many transport protocols, each offering a different channel abstraction to application programs. Thus, the issue of delivering messages from host to host is completely separated from the issue of providing a useful process-to-process communication service.

A final attribute of the Internet architecture is that in order for a new protocol to be officially included in the architecture, there must be both a protocol specification and at least one (and preferably two) representative implementations of the specification.

---

2. What is the need for error detection? Explain with typical examples. Explain methods used for error detection and error correction. [APR/MAY-2015,2016,17]

(or)

Explain the CRC error detection mechanism (8 marks) [NOV/DEC -2012]

INTRODUCTION:

The bit errors are sometimes introduced into frames. This happens, for example, because of electrical interference or thermal noise. Hamming and Reed-Solomon codes are two notable examples that were developed for use in punch card readers, when storing data on magnetic disks, and in early core memories.

Detecting errors is only one part of the problem. The other part is correcting errors once detected. Two basic approaches can be taken when the recipient of a message detects an error. One is to notify the sender that the message was corrupted so that the sender can retransmit a copy of the message.

One of the most common techniques for detecting transmission errors is a technique known as the *cyclic redundancy check* (CRC).

The basic idea behind any error detection scheme is to add redundant information to a frame that can be used to determine if errors have been introduced. The goal of error detecting codes is to provide a high probability of detecting errors combined with a relatively low number of redundant bits.

TWO DIMENSIONAL PARITY

Two-dimensional parity is exactly what the name suggests. It is based on ―simple‖ (one-dimensional) parity, which usually involves adding one extra bit to a 7-bit code to balance the number of 1s in the byte. For example, odd parity sets the eighth bit to 1 if needed to give an odd number of 1s in the byte, and even parity sets the eighth bit to 1 if needed to give an even number of 1s in the byte. Two-dimensional parity does a similar calculation for each bit position across each of the bytes contained in the frame. This results in an extra parity byte for the entire frame, in addition to a parity bit for each byte.

The below figure illustrates how two-dimensional even parity works for an example frame containing 6 bytes of data. Notice that the third bit of the parity byte is 1 since there is an odd number of 1s in the third bit across the 6 bytes in the frame. It can be shown that two-dimensional parity catches all 1-, 2-, and 3-bit errors, and most 4-bit errors. In this case, 14 bits of redundant information is added to a 42-bit message, and yet it yields stronger protection against common errors than the –repetition code‖.

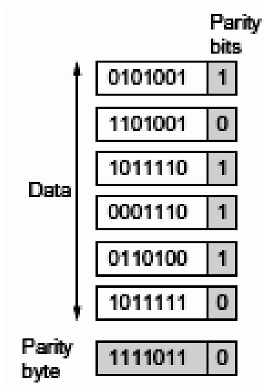


Fig: Two Dimensional Parity

INTERNET CHECKSUM

The idea behind the Internet checksum is very simple—you add up all the words that are transmitted and then transmit the result of that sum. The result is the checksum. The receiver performs the same calculation on the received data and compares the result with the received checksum. If any transmitted data, including the checksum itself, is corrupted, then the results will not match, so the receiver knows that an error occurred. The exact scheme used by the Internet protocols works as follows.

Consider the data being check summed as a sequence of 16-bit integers. Add them together using 16-bit ones complement arithmetic and then take the ones complement of the result. That 16-bit number is the checksum.

This algorithm scores well for using a small number of redundant bits—only 16 for a message of any length— but it does not score extremely well for strength of error detection. One reason it is adequate is that this checksum is the last line of defense in an end-to-end protocol

CYCLIC REDUNDANCY CHECK

The major goal in designing error detection algorithms is to maximize the probability of detecting errors using only a small number of redundant bits. Cyclic redundancy checks use some fairly powerful mathematics to achieve this goal. For example, a 32- bit CRC gives strong protection against common bit errors in messages that are thousands of bytes long.

CRC is based on binary division. In this a sequence of redundant bits, called CRC remainder is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second predetermined binary number. At its destination, the incoming data unit is divided by the same number. If at this step there is no reminder, the data unit is assumed to be intact and therefore accepted. A remainder indicates that the data unit has been changed in transit and therefore must be rejected.

Here, the remainder is the CRC. It must have exactly one less bit than the divisor, and appending it to the end of the data string must make the resulting bit sequence exactly divisible by the divisor.

First, a string of n-1 0s is appended to the data unit. The number of 0s is one less than the number of bits in the divisor which is n bits. Then the newly elongated data unit is divided by the divisor using a process called binary division. The remainder is CRC. The CRC is replaces the appended 0s at the end of the data unit.

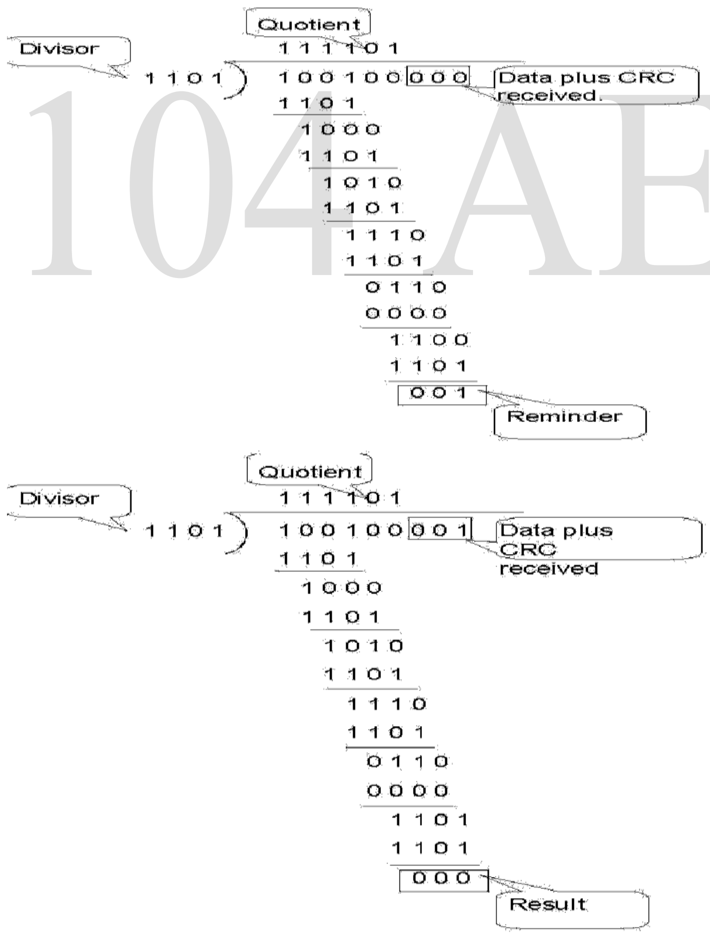


Fig: Cyclic Redundancy Check

The data unit arrives at the receiver first, followed by the CRC. The receiver treats whole string as the data unit and divides it by the same divisor that was used to find the CRC remainder. If the remainder is 0 then the data unit is error free. Otherwise it is having some error and it must be discarded.

3.Discuss the issues in Data link layer [NOV/DEC -2014] (or)
Discuss the framing technique used in HDLC. What is the effect of errors on this framing? (8 marks) [MAY/JUNE – 2013,17]

(or)

Describe how bit stuffing works in HDLC protocol (8marks) [NOV/DEC-2013]

FRAMING

It is the network adaptor that enables the nodes to exchange frames. When node A wishes to transmit a frame to node B, it tells its adaptor to transmit a frame from the node‘s memory. This results in a sequence of bits being sent over the link. The adaptor on node B then collects together the sequence of bits arriving on the link and deposits the corresponding frame in B‘s memory. Recognizing exactly what set of bits constitutes a frame—that is, determining where the frame begins and ends—is the central challenge faced by the adaptor

There are several ways to solve the framing problem. They are,

- **Byte-Oriented Protocols (BISYNC, PPP, DDCMP)**
- **Bit-Oriented Protocols (HDLC)**

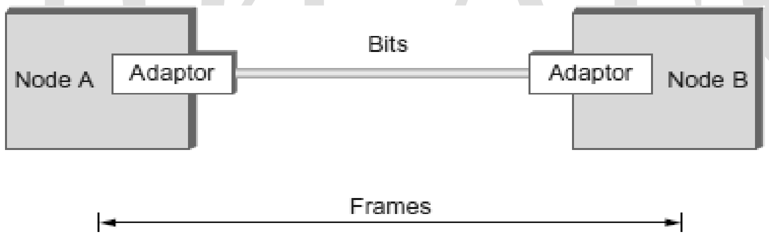


Fig: Bits flow between adaptors, frames between hosts.

BYTE ORIENTED PROTOCOLS (BISYNC, PPP, DDCMP):

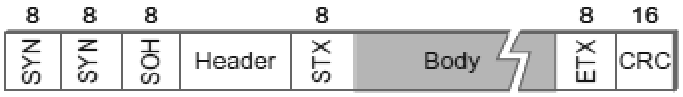
One of the oldest approaches to framing is to view each frame as a collection of bytes (characters) rather than a collection of bits. Such a *byte-oriented* approach is exemplified by older protocols such as the Binary Synchronous Communication (BISYNC) protocol developed by IBM in the late 1960s, and the Digital Data Communication Message Protocol (DDCMP) used in Digital Equipment Corporation‘s DECNET. The more recent and widely used Point-to-Point Protocol (PPP) provides another example of this approach.

Sentinel-Based Approaches (BISYNC, PPP)

In this approach it uses special characters called sentinel characters to indicate where frames start and end. This approach is called character stuffing because extra characters are inserted in the data portion of the frame.

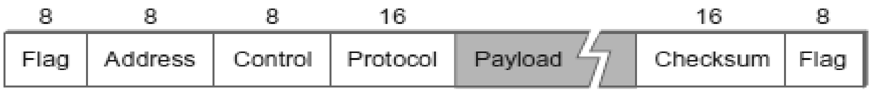
The below figure illustrates the BISYNC protocol‘s frame format. The packets are transmitted beginning with the leftmost field. BISYNC uses special characters known as *sentinel characters* to indicate where frames start and end. The beginning of a frame is denoted by sending a special SYN (synchronization) character. The data portion of the frame is then contained between two more special characters: STX (start of text) and ETX (end of text). The SOH (start of header) field serves much the same purpose as the STX field. The problem with the sentinel approach, of course, is that the ETX character might appear in the data portion of the frame. BISYNC overcomes this problem by ―escaping‖ the ETX character by preceding it with a DLE (data-link-escape) character whenever it appears in the body of a frame; the DLE character is also escaped (by preceding it with an extra DLE) in the frame body.

This approach is often called *character stuffing* because extra characters are inserted in the data portion of the frame. The frame format also includes a field labeled CRC, which is used to detect transmission errors. Finally, the frame contains additional header fields the link level reliable delivery algorithm.



The more recent Point-to-Point Protocol (PPP), which is commonly used to carry Internet Protocol packets over various sorts of point-to-point links, is similar to BISYNC in that it also uses sentinels and character stuffing. The format for a PPP frame is given in the figure below. The special start-of-text character, denoted as the Flag field 01111110. The Address and Control fields usually contain default values and so are uninteresting. The Protocol field is used for demultiplexing; it identifies the high-level protocol such as IP or IPX. The frame payload size can be negotiated, but it is 1500 bytes by default.

The Checksum field is either 2 (by default) or 4 bytes long. The PPP frame format is unusual in that several of the field sizes are negotiated rather than fixed. This negotiation is conducted by a protocol called the Link Control Protocol (LCP).

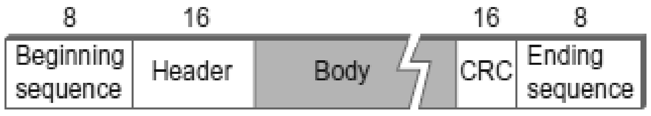


Bit-Oriented Protocols (HDLC)

A bit-oriented protocol is not concerned with byte boundaries—it simply views the frame as a collection of bits. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. HDLC denotes both the beginning and the end of a frame with the distinguished bit sequence 01111110.

Bit stuffing in the HDLC protocol works as follows. On the sending side, any time five consecutive 1s have been transmitted from the body of the message (i.e., excluding when the sender is trying to transmit the distinguished 01111110 sequence), the sender inserts a 0 before transmitting the next bit.

On the receiving side, should five consecutive 1s arrive, the receiver makes its decision based on the next bit it sees (i.e., the bit following the five 1s). If the next bit is a 0, it must have been stuffed, and so the receiver removes it. If the next bit is a 1, then one of two things is true: Either this is the end-of-frame marker or an error has been introduced into the bit stream. By looking at the *next* bit, the receiver can distinguish between these two cases. If it sees a 0 (i.e., the last 8 bits it has looked at are 01111110), then it is the end-of-frame marker; if it sees a 1 (i.e., the last 8 bits it has looked at are 01111111), then there must have been an error and the whole frame is discarded. In the latter case, the receiver has to wait for the next 01111110 before it can start receiving again, and, as a consequence, there is the pote ntial that the receiver will fail to receive two consecutive frames..



4. **Discuss briefly about Link level flow control (8/16 marks) [NOV/DEC -12,16]**

INTRODUCTION:

An acknowledgment is a small control frame that a protocol sends back to its peer saying that it has received an earlier frame. The receipt of an acknowledgment indicates to the sender of the original frame that its frame was successfully delivered. If the sender does not receive an acknowledgment after a reasonable amount of time, then it *retransmits* the original frame. This action of waiting a reasonable amount of time is called a *timeout*. The general strategy of using acknowledgments and timeouts to implement reliable delivery is sometimes called *automatic repeat request*.

There are three different ARQ algorithms using generic language.

- Stop-and-Wait
- Sliding Window
- Concurrent Logical Channels

STOP-AND-WAIT

The simplest ARQ scheme is the stop-and-wait algorithm. The idea of stop-and-wait is straightforward: After transmitting one frame, the sender waits for an acknowledgment before transmitting the next frame. If the acknowledgment does not arrive after a certain period of time, the sender times out and retransmit the original frame.

The figure below illustrates four different scenarios that result from this basic algorithm. The sending side is represented on the left, the receiving side is depicted on the right, and time flows from top to bottom.

Figure (a) shows the situation in which the ACK is received before the timer expires; Figure (b) and (c) show the situation in which the original frame and the ACK, respectively, are lost; and Figure (d) shows the situation in which the timeout fires too soon

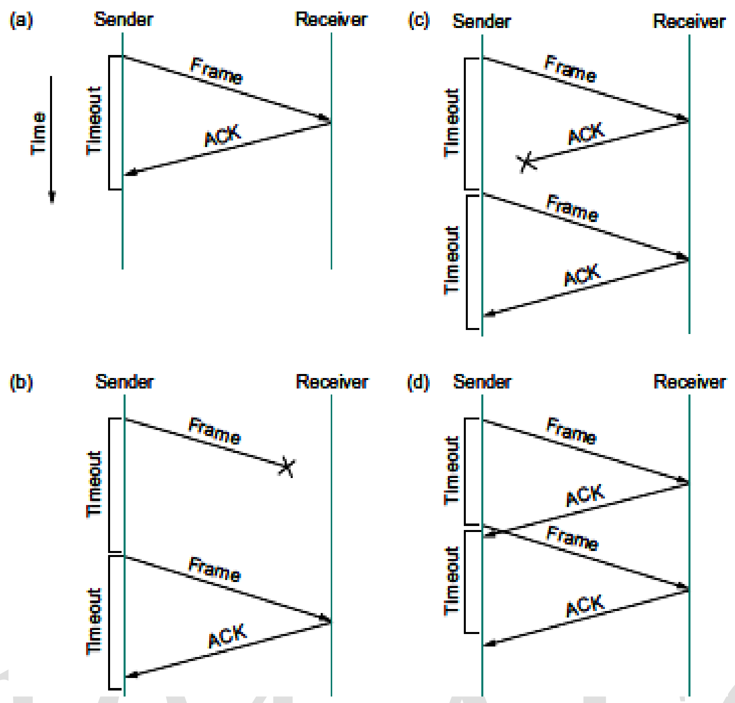


Fig: Timeline showing four different scenarios for the stop-and-wait algorithm. (a) The ACK is received before the timer expires; (b) the original frame is lost; (c) the ACK is lost; (d) the timeout fires too soon

Suppose the sender sends a frame and the receiver acknowledges it, but the acknowledgment is either lost or delayed in arriving. This situation is illustrated in timelines (c) and (d) of the above figure. In both cases, the sender time out and retransmits the original frame, but the receiver will think that it is the next frame, since it correctly received and acknowledged the first frame. This has the potential to cause duplicate copies of a frame to be delivered. To address this problem, the header for a stop-and-wait protocol usually includes a 1-bit sequence number—that is, the sequence number can take on the values 0 and 1—and the sequence numbers used for each frame alternate, as illustrated in the figure below.

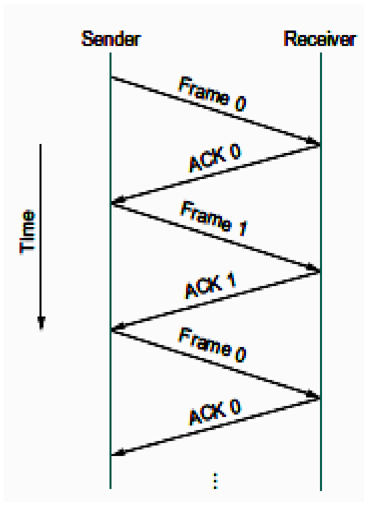


Fig: Timeline for stop-and-wait with 1-bit sequence number

The main shortcoming of the stop-and-wait algorithm is that it allows the sender to have only one outstanding frame on the link at a time, and this may be far below the link‘s capacity. Consider, for example, a 1.5-Mbps link with a 45-ms round-trip time. This link has a delay× bandwidth product of 67.5 Kb, or approximately 8 KB. Since the sender can send only one frame per RTT, and assuming a frame size of 1 KB, this implies a maximum sending rate of

Bits Per Frame÷Time Per Frame

= 1024×8÷0.045

= 182 kbps

SLIDING WINDOW

In this method, the sender can transmit several frames before receiving an acknowledgment. The receiver acknowledges only some of the frames, using a single ACK to confirm the receipt of multiple data frames. The timeline is illustrated in the figure below.

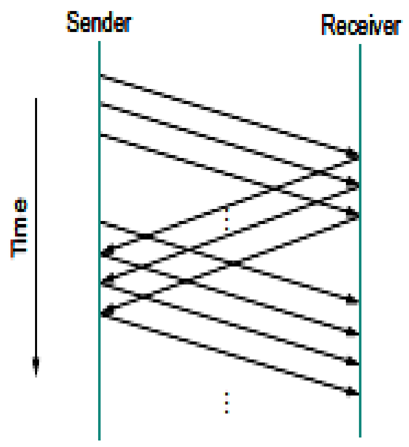


Fig: Timeline for the sliding window algorithm.

The Sliding Window Algorithm

The sliding window algorithm works as follows. First, the sender assigns a *sequence number*, denoted SeqNum, to each frame. Consider that SeqNum is not implemented by a finite-size header field and instead assume that it can grow infinitely large. The sender maintains three variables: The *send window size,* denoted SWS, gives the upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit; LAR denotes the sequence number of the *last acknowledgment received*; and LFS denotes the sequence number of the *last frame sent*. The sender also maintains the following invariant:

$$\text{LFS} - \text{LAR} \leq \text{SWS}$$

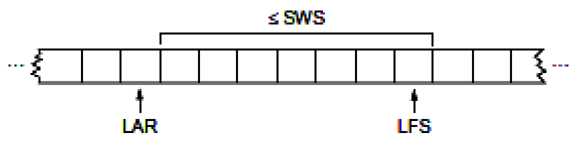


Fig: Sliding Window on sender

When an acknowledgment arrives, the sender moves LAR to the right, thereby allowing the sender to transmit another frame. Also, the sender associates a timer with each frame it transmits, and it retransmits the frame should the timer expire before an ACK is received. The receiver maintains the following three variables: The *receive window size*, denoted RWS, gives the upper bound on the number of out-of-order frames that the receiver is willing to accept; LAF denotes the sequence number of the *largest acceptable frame*; and LFR denotes the sequence number of the *last frame received*. The receiver also maintains the following invariant:

$$\text{LAF} - \text{LFR} \leq \text{RWS}$$

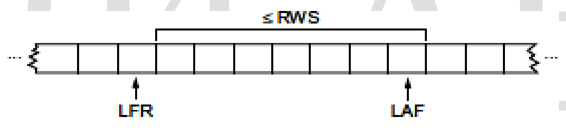


Fig: Sliding Window on receiver

When a frame with sequence number SeqNum arrives, the receiver takes the following action. If SeqNum ≤ LFR or SeqNum > LAF, then the frame is outside the receiver‘s window and it is discarded. If LFR <SeqNum ≤ LAF, then the frame is within the receiver‘s window and it is accepted. Now the receiver needs to decide whether or not to send an ACK. Let SeqNumToAck denote the largest sequence number not yet acknowledged, such that all frames with sequence numbers less than or equal to SeqNumToAck have been received. The receiver acknowledges the receipt of SeqNumToAck, even if higher numbered packets have been received. This acknowledgment is said to be cumulative. It then sets LFR = SeqNumToAck and adjusts LAF = LFR+RWS.

Another variation on this scheme would be to use *selective acknowledgments*. That is, the receiver could acknowledge exactly those frames it has received rather than just the highest numbered frame received in order.

The following routine illustrate how we might implement the sending and receiving sides of the sliding window algorithm. The routines are taken from a working protocol named ,Sliding Window Protocol (SWP). First, the frame header is very simple: It contains a sequence number (SeqNum) and an acknowledgment number (AckNum). It also contains a Flags field that indicates whether the frame is an ACK or carries data.

---

5. How frame order and flow control is achieved using the data link layer?

[MAY/JUNE -2014,NOV/DEC-2015]

Frame Order and Flow Control

The sliding window protocol is perhaps the best known algorithm in computer networking. It can be used to serve three different roles.

- The first role is the one is to reliably deliver frames across an unreliable link.
- The second role that the sliding window algorithm can serve is to preserve the order in which frames are transmitted.
- The third role that the sliding window algorithm sometimes plays is to support *flow control*—a feedback mechanism by which the receiver is able to throttle the sender. Such a mechanism is used to keep the sender from over-running the receiver—that is, from transmitting more data than the receiver is able to process. This is usually accomplished by augmenting the sliding window protocol so that the receiver not only acknowledges frames it has received but also informs the sender of how many frames it has room to receive. The number of frames that the receiver is capable of receiving corresponds to how much free buffer space it has. As in the case of ordered delivery, we need to make sure that flow control is necessary at the link level before incorporating it into the sliding window protocol.

Concurrent Logical Channels

The data link protocol used in the ARPANET provides an interesting alternative to the sliding window protocol, in that it is able to keep the pipe full while still using the simple stop-and-wait algorithm.

The frames sent over a given link are not kept in any particular order. The protocol also implies nothing about flow control. The idea underlying the ARPANET protocol, which we refer to as *concurrent logical channels*, is to multiplex several logical channels onto a single point-to-point link and to run the stop-and-wait algorithm on each of these logical channels. There is no relationship maintained among the frames sent on any of the logical channels.

The sender keeps 3 bits of state for each channel: a boolean, saying whether the channel is currently busy; the 1-bit sequence number to use the next time a frame is sent on this logical channel; and the next sequence number to expect on a frame that arrives on this channel. When the node has a frame to send, it uses the lowest idle channel, and otherwise it behaves just like stop-and-wait.

In practice, the ARPANET supported 8 logical channels over each ground link and 16 over each satellite link.

UNIT 2

MEDIA ACCESS AND INTERNETWORKING

PART – B

1. Explain in detail about the access method and frame format used in Ethernet (IEEE 802.3). [APR/MAY-2015]

Describe the transmitter algorithm implemented at the sender side of the Ethernet protocol. Why should Ethernet frame should be 512 bytes long? [NOV/DEC-2013]

Ethernet is a multiple-access network, meaning that a set of nodes sends and receives frames over a shared link. The ―carrier sense‖ in CSMA/CD means that all the nodes can distinguish between an idle and a busy link, and ―collision detect‖ means that a node listens as it transmits and can therefore detect when a frame it is transmitting has interfered (collided) with a frame transmitted by another node.

Physical Properties

Ethernet segments were originally implemented using coaxial cable of length up to 500 m.A *transceiver*, a small device directly attached to the tap, detected when the line was idle and drove the signal when the host was transmitting. It also received incoming signals. The transceiver, in turn, connected to an Ethernet adaptor, which was plugged into the host. Multiple Ethernet segments can be joined together by *repeaters*.

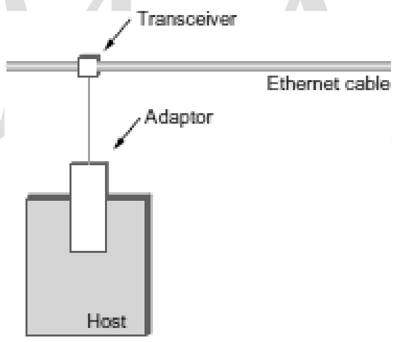


Fig: Ethernet transceiver and receiver

A repeater is a device that forwards digital signals, much like an amplifier forwards analog signals. Repeaters understand only bits, not frames; however, no more than four repeaters could be positioned between any pair of hosts, meaning that a classical Ethernet had a total reach of only 2500 m.

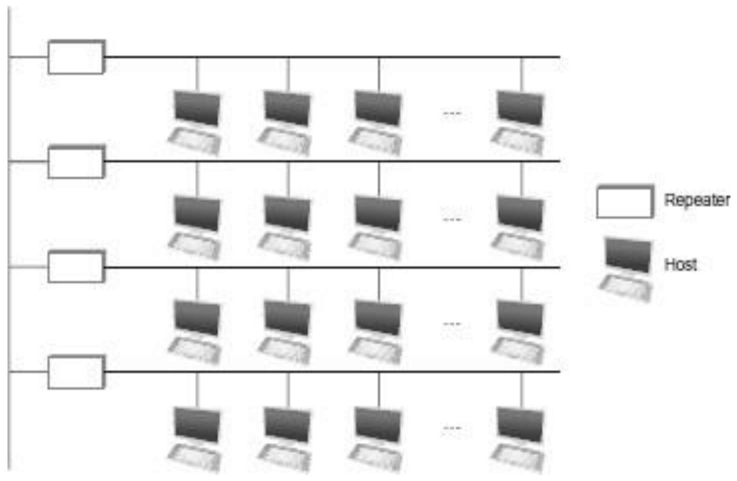


Fig: Ethernet repeater

Access Protocol

It is the algorithm that controls access to a shared Ethernet link. This algorithm is commonly called the Ethernet‘s *media access control* (MAC). It is typically implemented in hardware on the network adaptor.

Frame Format

Each Ethernet frame is defined by the format given in the figure below**.** The 64-bit preamble allows the receiver to synchronize with the signal; it is a sequence of alternating 0s and 1s. Both the source and destination hosts are identified with a 48-bit address.

The packet type field serves as the demultiplexing key; it identifies to which of possibly many higher level protocols this frame should be delivered. Each frame contains up to 1500 bytes of data. Minimally, a frame must contain at least 46 bytes of data, even if this means the host has to pad the frame before transmitting it. The reason for this minimum frame size is that the frame must be long enough to detect a collision; Finally, each frame includes a 32-bit CRC.

The Ethernet is a bit-oriented framing protocol. An Ethernet frame has a 14-byte header: two 6-byte addresses and a 2-byte type field. The sending adaptor attaches the preamble and CRC before transmitting, and the receiving adaptor removes them.

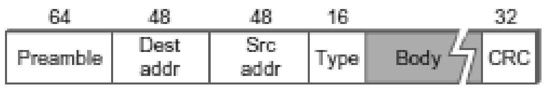


Fig: Ethernet Frame format

Addresses

Ethernet addresses are typically printed as a sequence of six numbers separated by colons. Each number corresponds to 1 byte of the 6-byte address and is given by a pair of hexadecimal digits, one for each of the 4-bit nibbles in the byte; leading 0s are dropped. For example, 8:0:2b:e4:b1:2 is the human-readable representation of Ethernet address **00001000 00000000 00101011 11100100 10110001 00000010**

To ensure that every adaptor gets a unique address, each manufacturer of Ethernet devices is allocated a different prefix that must be prepended to the address on every adaptor they build.

A given manufacturer then makes sure the address suffixes it produces are unique. Each frame transmitted on an Ethernet is received by every adaptor connected to that Ethernet. Each adaptor recognizes those frames addressed to its address and passes only those frames on to the host.

An Ethernet address consisting of all 1s is treated as a *broadcast* address; all adaptors pass frames addressed to the broadcast address up to the host. Similarly, an address that has the first bit set to 1 but is not the broadcast address is called a *multicast* address.. To summarize, an Ethernet adaptor receives all frames and accepts

- Frames addressed to its own address
- Frames addressed to the broadcast address
- Frames addressed to a multicast address, if it has been instructed to listen to that address

Transmitter Algorithm

The transmitter algorithm is defined as follows.

When the adaptor has a frame to send and the line is idle, it transmits the frame immediately; there is no negotiation with the other adaptors. The upper bound of 1500 bytes in the message means that the adaptor can occupy the line for only a fixed length of time.

When an adaptor has a frame to send and the line is busy, it waits for the line to go idle and then transmits immediately.7 The Ethernet is said to be a *1-persistent* protocol because an adaptor with a frame to send transmits with probability 1 whenever a busy line goes idle. In general, a *p-persistent* algorithm transmits with probability $0 \leq p \leq 1$ after a line becomes idle and defers with probability $q = 1-p$. The reasoning behind choosing a $p < 1$ is that there might be multiple adaptors waiting for the busy line to become idle, and we don‘t want all of them to begin transmitting at the same time. If each adaptor transmits immediately with a probability of, say, 33%, then up to three adaptors can be waiting to transmit and the odds are that only one will begin transmitting when the line becomes idle.

In Ethernet, because there is no centralized control it is possible for two (or more) adaptors to begin transmitting at the same time, either because both found the line to be idle or because both had been waiting for a busy line to become idle. When this happens, the two (or more) frames are said to *collide* on the network. Each sender, because the Ethernet supports collision detection, is able to determine that a collision is in progress. At the moment an adaptor detects that its frame is colliding with another, it first makes sure to transmit a 32-bit jamming sequence and then stops the transmission. Thus, a transmitter will minimally send 96 bits in the case of a collision: 64-bit preamble plus 32-bit jamming sequence.

One way that an adaptor will send only 96 bits—which is sometimes called a *runt frame*—is if the two hosts are close to each other. Had the two hosts been farther apart, they would have had to transmit longer, and thus send more bits, before detecting the collision.

Ethernet frame must be at least 512 bits (64 bytes) long: 14 bytes of header plus 46 bytes of data plus 4 bytes of CRC because the farther apart two nodes are, the longer it takes for a frame sent by one to reach the other, and the network is vulnerable to a collision during this time.

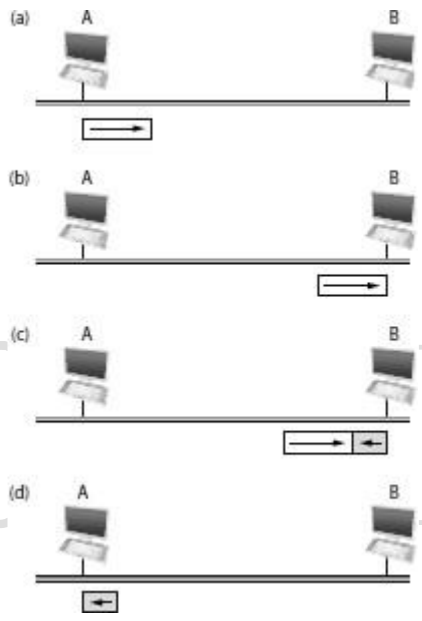


Fig: Transmitter Algorithm

The above figure illustrates the worst-case scenario, where hosts A and B are at opposite ends of the network. Suppose host A begins transmitting a frame at time t, as shown in (a). It takes it one link latency (let‘s denote the latency as d) for the frame to reach host B. Thus, the first bit of A‘s frame arrives at B at time t+d, as shown in (b). Suppose an instant before host A‘s frame arrives (i.e., B still sees an idle line), host B begins to transmit its own frame. B‘s frame will immediately collide with A‘s frame, and this collision will be detected by host B (c). Host B will send the 32-bit jamming sequence, as described above. (B‘s frame will be a runt.)

Unfortunately, host A will not know that the collision occurred until B‘s frame reaches it, which will happen one link latency later, at time t+2×d, as shown in (d). Host A must continue to transmit until this time in order to detect the collision. In other words, host A must transmit for 2×d. to be sure that it detects all possible collisions.

Considering that maximally configured Ethernet is 2500 m long, and that there may be up to four repeaters between any two hosts, the round-trip delay has been determined to be 51.2 µs, which on a 10-Mbps Ethernet corresponds to 512 bits. The other way to look at this situation is that we need to limit the Ethernet‘s maximum latency to a fairly small value (e.g., 51.2 µs) for the access algorithm to work; hence, an Ethernet‘s maximum length must be something on the order of 2500 m.

Experience with Ethernet

First an Ethernet is extremely easy to administer and maintain: There were no switches in the original Ethernets, no routing or configuration tables to be kept up-to-date, and it is easy to add a new host to the network. It is hard to imagine a simpler network to administer.

Second, it is inexpensive: Cable is cheap, and the only other cost is the network adaptor on each host.

---

2. Discuss the MAC layer functions of IEEE 802.11 and briefly define key requirements of wireless LAN. [APR/MAY-2015,MAY/JUNE-2016,17]

Explain how the hidden node and exposed node problem is addressed in 802.11 [NOV/DEC-2013,15]

IEEE 802.11 defines two MAC sublayers: the distributed coordination function (DCF) and point coordination function (PCF).

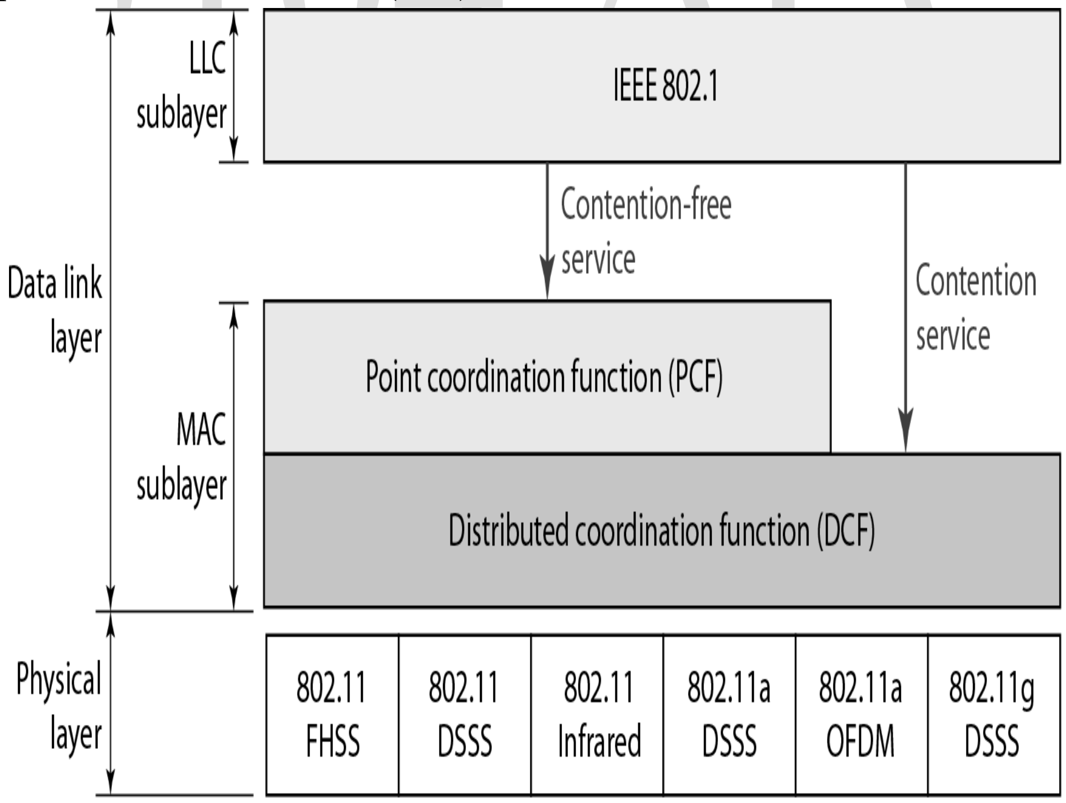


Fig: Layer of IEEE 802.11

Distributed Coordination Function

One of the two protocols defined by IEEE at the MAC sublayer is called the distributed coordination function (DCF). DCF uses *CSMAICA* as the access method.

I. 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.

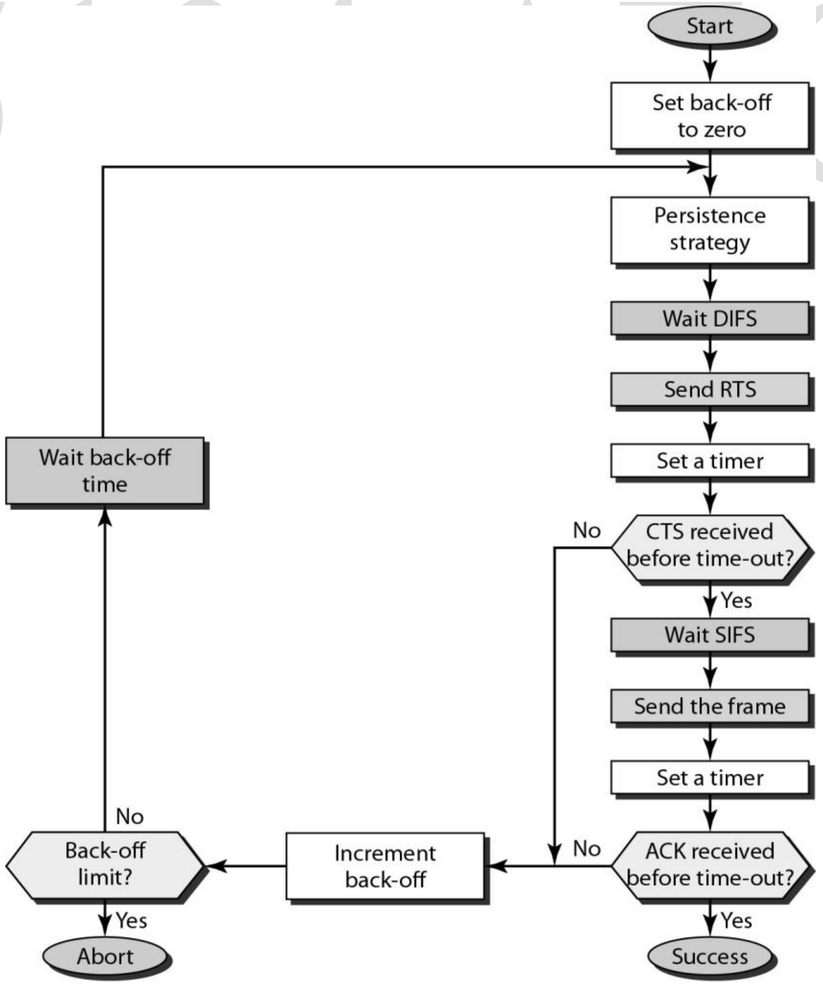
b. After the station is found to be idle, the station waits for a period of time called the distributed interframe space (DIFS); then the station sends a control frame called the request to send (RTS).

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.

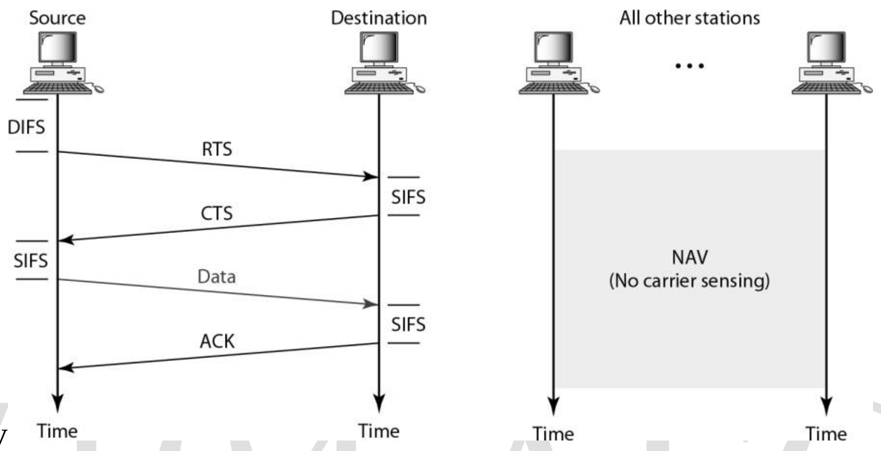
4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received..

CSMA/CA FLOWCHART



NAV 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.



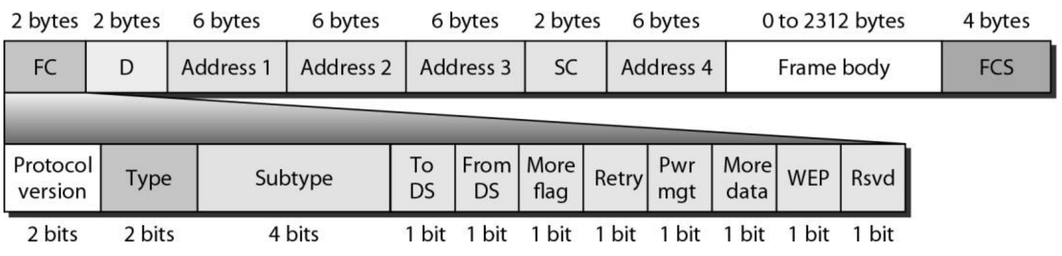
nav

POINT COORDINATION FUNCTION

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.

FRAME FORMAT

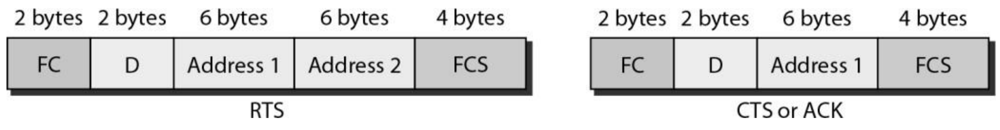


Field	*Explanation*
Version	Current version is 0
Type	Type of information: management (00), control (01), or data (10)
Subtype	Subtype of each type (see Table 14.2)
To DS	Defined later
From DS	Defined later
More flag	When set to 1, means more fragments
Retry	When set to 1, means retransmitted frame
Pwr 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

Frame Format

The MAC layer frame consists of nine fields.

- **Frame control (FC).** The FC field is 2 bytes long and defines the type of frame and some control information. In all frame types except one, this field defines the duration of the transmission that is used to set the value of NAV. 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.
- **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

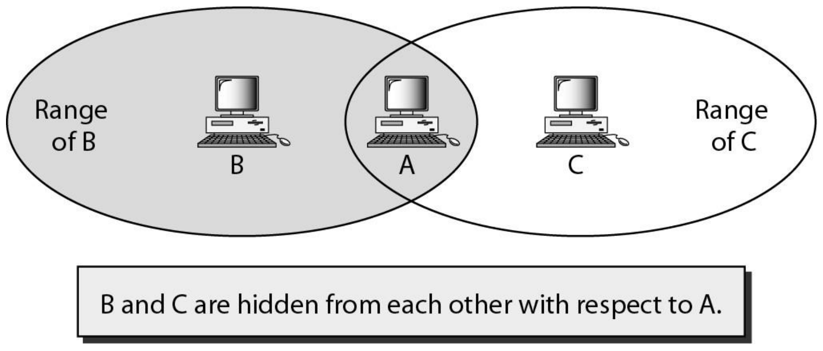


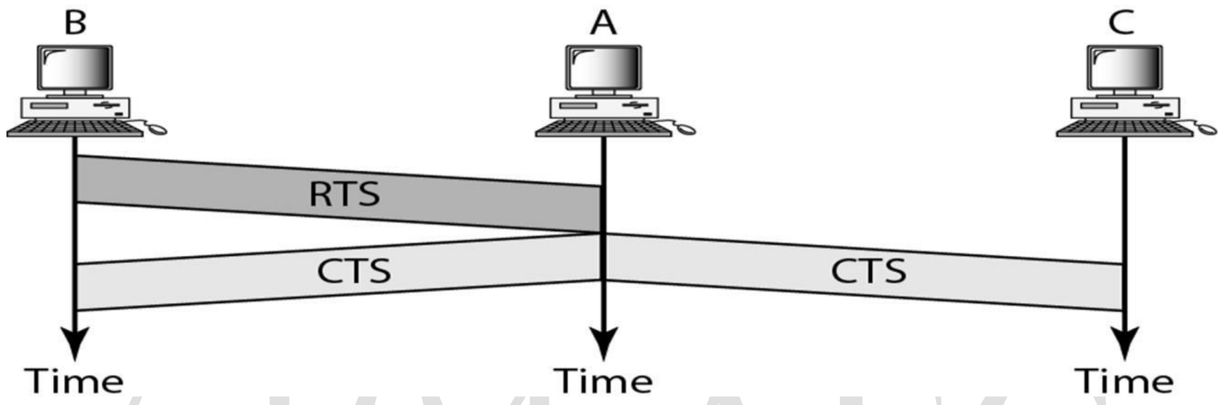
Frame Types

A wireless LAN defined by IEEE 802.11 has three categories of frames: management frames, control frames, and data frames.

Management frames are used for the initial communication between stations and access points Control Frames Control frames are used for accessing the channel and acknowledging frames. Data Frames are used for carrying data and control information.

Hidden Station Problem

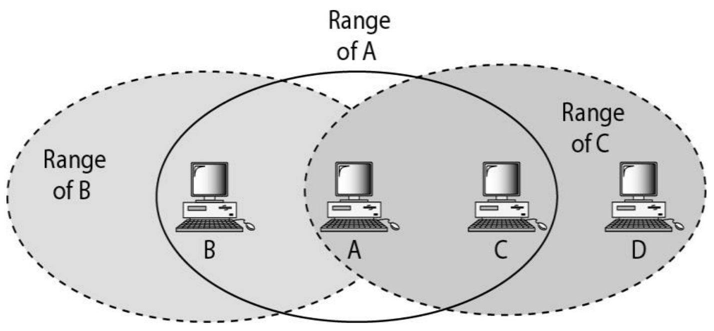




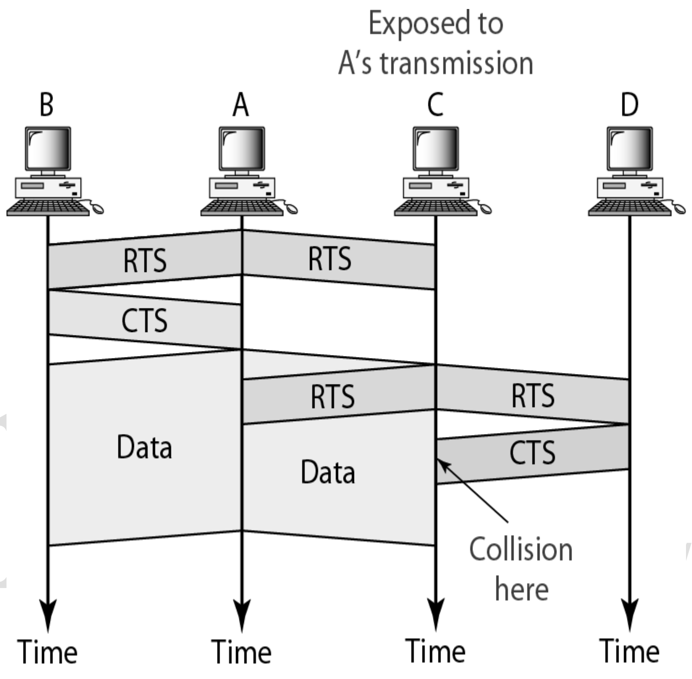
Station B has a transmission range shown by the left oval (sphere in space);every station in this range can hear any signal transmitted by station B. Station C has a transmission range shown by the right oval (sphere in space); every station located in this range can hear any signal transmitted by C. 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.

Exposed station problem

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.



C is exposed to transmission from A to B.



3.Discuss the IP addressing methods [MAY/JUNE-2014] & [APR/MAY-2011]

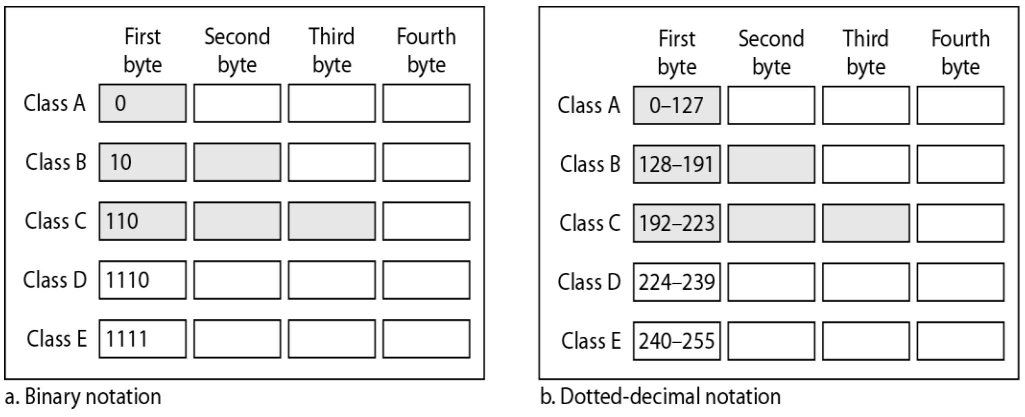
IPv4 ADDRESSES

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.

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.

IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296. 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.



Classes and Blocks

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

Netid and Hostid

In classful addressing, an IP address in class A, B, or C is divided into netid and hostid. 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.

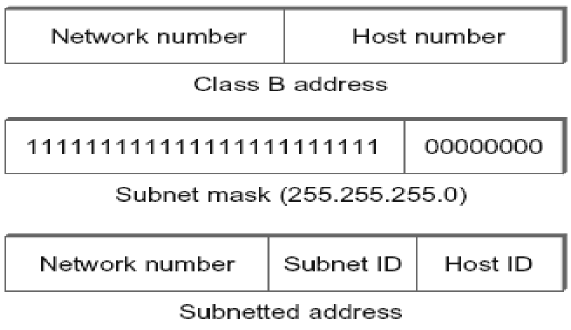
Mask

Although the length of the netid and hostid (in bits) is predetermined in classful addressing, we can also use a mask, a 32-bit number made of contiguous Is followed by contiguous as.

Class	*Binary*	*Dotted-Decimal*	*CIDR*
A	**11111111** 00000000 00000000 00000000	**255**.0.0.0	/8
B	**11111111 11111111** 00000000 00000000	**255.255**.0.0	/16
C	**11111111 11111111 11111111** 00000000	**255.255.255**.0	/24

Subnetting

The idea is to take a single IP network number and allocate the IP addresses with that network number to several physical networks, which are now referred to as *subnets*



The mechanism by which a single network number can be shared among multiple networks involves configuring all the nodes on each subnet with a *subnet mask*.. The subnet mask enables us to introduce a *subnet number*; all hosts on the same physical network will have the same subnet number, which means that hosts may be on different physical networks but share a single network number.

Classless addressing

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.
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.

4.Write short notes on ARP and DHCP. [MAY/JUNE-14,16,17, NOV/DEC-12] Address Translation (ARP).

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. 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.
The fields are 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 080016, 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) and ARP 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.

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.

Host Configuration (DHCP)

The Dynamic Host Configuration Protocol (DHCP) has been devised to provide static and dynamic address allocation that can be manual or automatic.

Static Address Allocation.

A DHCP server has a database that statically binds physical addresses to IP addresses.

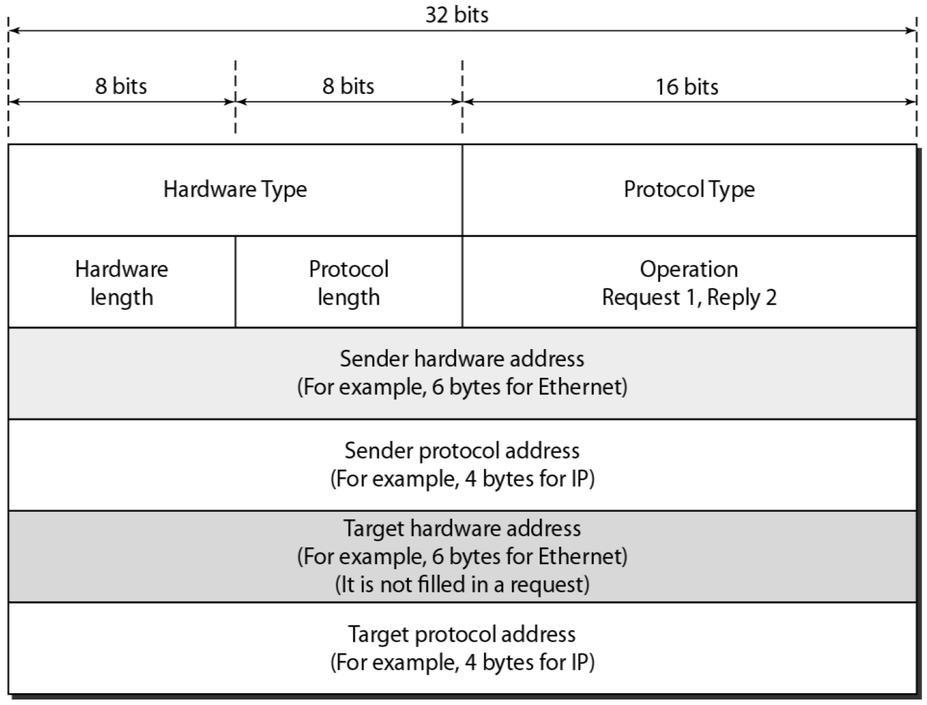

32 bits
8 bits
8 bits
16 bits
Hardware Type
Protocol Type
Hardware length
Protocol length
Operation Request 1, Reply 2
Sender hardware address (For example, 6 bytes for Ethernet)
Sender protocol address (For example, 4 bytes for IP)
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled in a request)
Target protocol address (For example, 4 bytes for IP)

Fig: ARP

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 . 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.

The below figure shows the format of a DHCP message. The message is actually sent using a protocol called the *User Datagram Protocol* (UDP) that runs over IP. UDP is

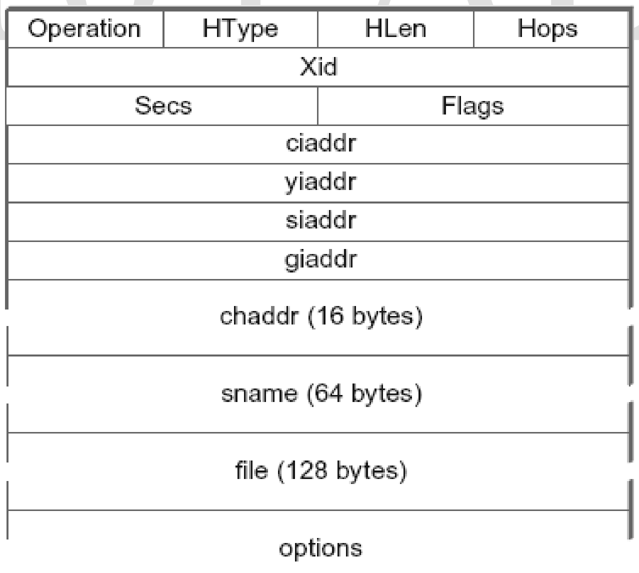


Fig: Dynamic Address Allocation.

5.Describe with example how CIDR addresses the two scaling concerns in the internet. Write in detail about ICMP [NOV/DEC-2013, MAY/JUNE-2016,17]

Classless Addressing

CIDR takes the subnetting idea to its logical conclusion by essentially doing away with address classes altogether.

Subnetting only allows us to split a classful address among multiple subnets, while CIDR allows us to coalesce several classful addresses into a single ―supernet.‖ Even though subnetting can help us to assign addresses carefully, it does not get around the fact that any organization with more than 255 hosts, or an expectation of eventually having that many, wants a class B address.

For any organization with at least 256 hosts, we can guarantee an address utilization of at least 50%, and typically much more. This solution, however, raises a problem that is at least as serious: excessive storage requirements at the routers.

CIDR, tries to balance the desire to minimize the number of routes that a router needs to know against the need to hand out addresses efficiently. To do this, CIDR helps us to *aggregate* routes.

CIDR requires a new type of notation to represent network numbers, or *prefixes* as they are known, because the prefixes can be of any length. The convention is to place a /X after the prefix, where X is the prefix length in bits

The ability to aggregate routes at the edge of the network as we have just seen is only the first step.

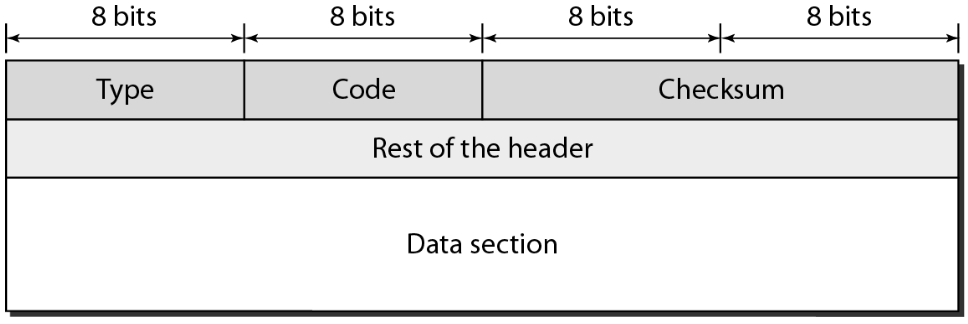
CIDR means that prefixes may be of any length, from 2 to 32 bits. Furthermore, it is sometimes possible to have prefixes in the forwarding table that ―overlap,‖ in the sense that some addresses may match more than one prefix.

For example, we might find both 171.69 (a 16-bit prefix) and 171.69.10 (a 24-bit prefix) in the forwarding table of a single router. In this case, a packet destined to, say, 171.69.10.5 clearly matches both prefixes. The rule in this case is based on the principle of ―longest match‖; that is, the packet matches the longest prefix, which would be 171.69.10 in this example.

On the other hand, a packet destined to 171.69.20.5 would match 171.69 and *not* 171.69.10, and in the absence of any other matching entry in the routing table 171.69 would be the longest match.

ICMP

The IP protocol has no error-reporting or error-correcting mechanism. The IP protocol also lacks a mechanism for host and management queries.The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protoco1



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

An ICMP message has an 8-byte header and a variable-size data section. The first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field. 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.
ICMP always reports error messages to the original source.

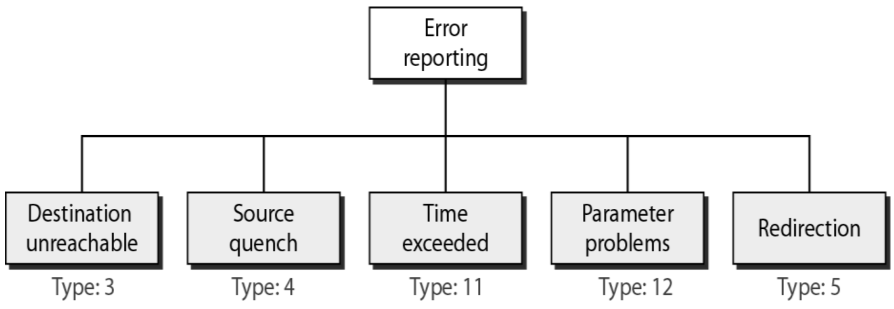


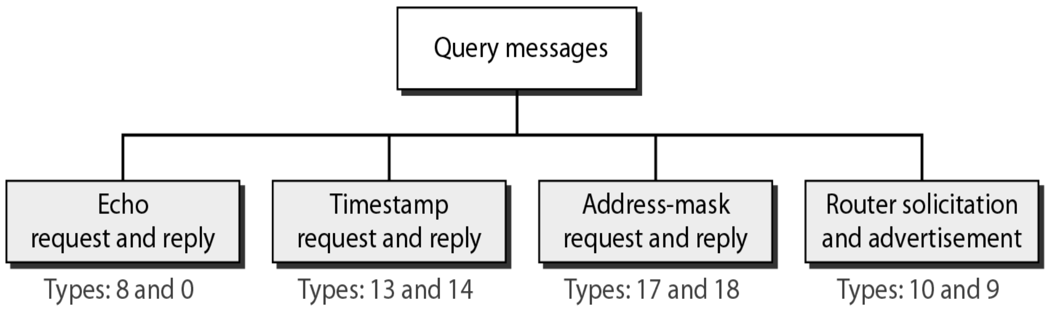
Fig: Error messages

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 ICMP 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.

Query

ICMP can diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages,

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



The echo-request and echo-reply messages can be used to determine if there is communication at the IP level

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

6. Briefly define key requirements of Wireless LAN. [APRIL/MAY – 2015]

The wireless LAN requirements should address the following elements:

- **Coverage and mobility—**Specifying the *coverage*, or area where the end users will operate the wireless devices, helps the designer determine the number and location of wireless LAN access points. *Mobility* requirements derive from the movement of users through the coverage areas. It's generally best to illustrate the coverage areas on drawings of the facility or building.
- **Application requirements—**These specify the transmission needs of the software that will operate over the wireless LAN. For example, warehouse-management system software requires the transmission of relatively low-bandwidth bar-code information between wireless handheld bar-code scanners and a host computer. A desktop video application, however, requires the transmission of real-time video signals. How frequently you expect to use these different information types helps determine the design specifications for your data transmission rate and throughput.
- **Number of users—**Simply put, this is the number of devices that require access to the wireless LAN. Be sure to allow for future expansion.
- **End-user device types—**Included in this category are laptop computers, bar-code data collectors, and mobile patient monitors. You should identify the available physical interfaces—such as PC card, PCI, ISA, or USB—for each device.
- **Battery longevity—**For mobile and portable applications, specify the length of time that the end-user devices need to operate on a set of batteries. This information indicates whether wireless LAN components need to support power-management functions.
- **System interfaces—**Most likely, the wireless LAN will have to interface with existing systems such as Ethernet networks, applications, and databases. The system interface requirements describe the architectures and communications protocols of these systems.
- **Information security requirements, or the level of protection that your data needs from particular threats**—The degree of needed security depends on the severity of the consequences that the organization faces if data is lost. These requirements determine whether you should include mechanisms such as encryption.
- **Environment—**Conditions such as room temperature and humidity, construction materials, floor space, and presence and intensity of electromagnetic waves could affect the operation of the system. In most cases, you should perform

a site survey to inspect the facility and evaluate the presence of potential radio-frequency interference.

- **Schedule—**Identify any circumstances that will affect the schedule, such as the availability of funds, an urgency to see a return on investment, the availability of project team members, and the interdependency between this project and others. Spell out the schedule requirements so that the team knows the time frames it must work within.
- **Budget—**An organization may have a limited amount of money to spend on the wireless LAN. If you know the budget constraints, it's best to identify them in the requirements.

UNIT 3 ROUTING

PART -B

1. **Discuss briefly about RIP and OSPF. [APR/MAY-2015],[MAY/JUNE-2012],[MAY/JUNE-2014,2016,17]& [NOV/DEC2014,2015,16]**

Explain the shortest path algorithm with suitable illustration.

Intra-domain routing

- Distance vector routing (eg. RIP)
- Link state routing (eg.OSPF)

Routing:

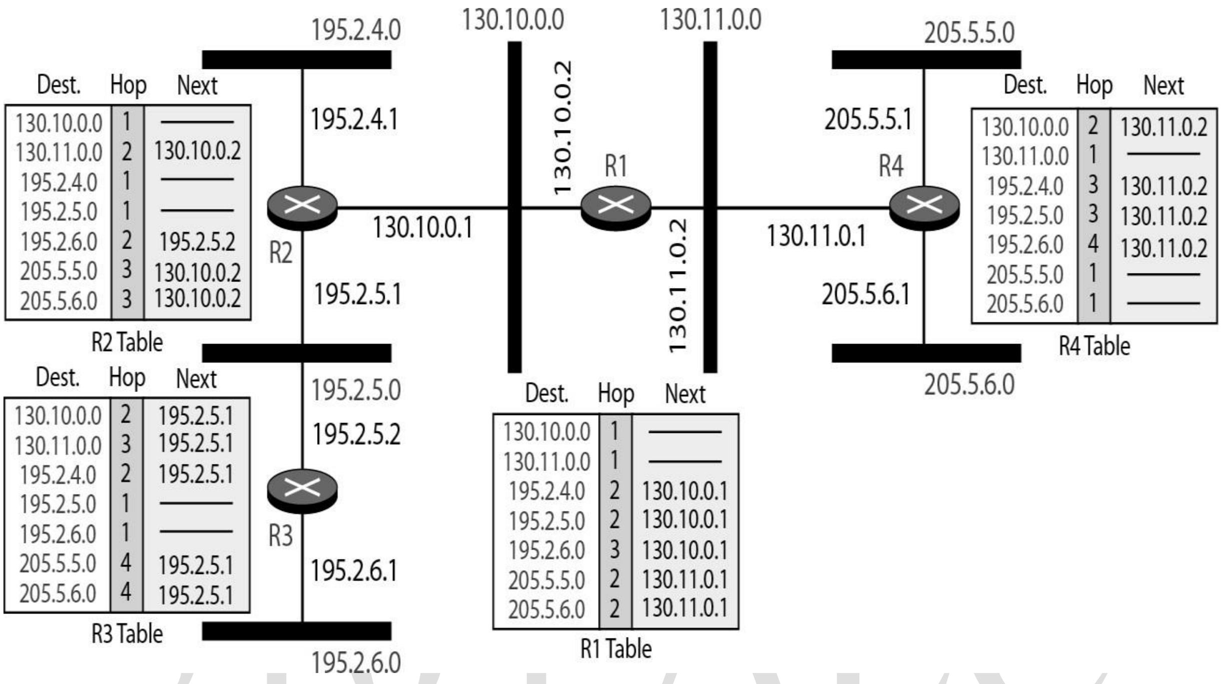
The process by which nodes exchange topological information to build correct forwarding tables are said to be *routing*.

Routing Information Protocol (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.
6. Routers running RIP send their advertisements every 30 seconds.

Figure 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 Rl. The table has seven entries to show how to reach each network in the autonomous system. Router Rl 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 Rl 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 Rl 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



R2 Table

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

R3 Table

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

R1 Table

Dest.	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

R4 Table

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	——

Fig: RIP

RIP V2 PACKET FORMAT

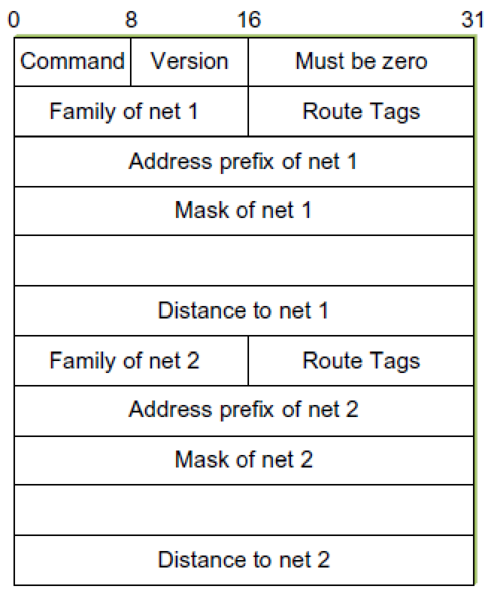


Fig: Rip V2 Packet Format

The Open Shortest Path First Protocol (OSPF)

OSPF have quite a number of features, including the following:

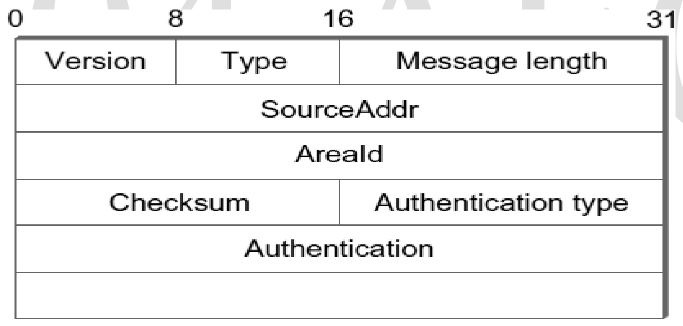
Authentication of routing messages

OSPF used a simple 8-byte password for authentication. This is not a strong enough form of authentication to prevent dedicated malicious users, but it alleviates some problems caused by mis configuration or casual attacks.

Additional hierarchy OSPF introduces another layer of hierarchy into routing by allowing a domain to be partitioned into *areas*. This means that a router within a domain does not necessarily need to know how to reach every network within that domain

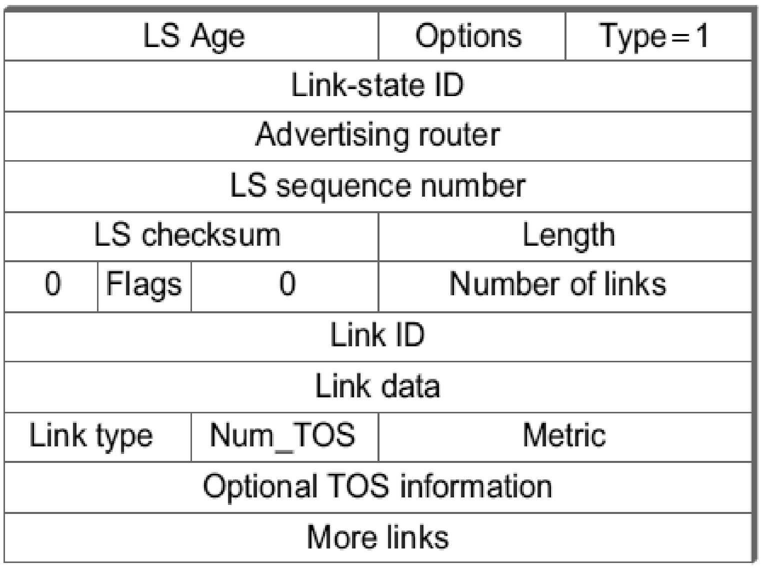
Load balancing OSPF allows multiple routes to the same place to be assigned the same cost and will cause traffic to be distributed evenly over those routes, thus making better use of available network capacity.

The Version field is currently set to 2, and the Type field may take the values 1 through 5. The SourceAddr identifies the sender of the message, and the AreaId is a 32-bit identifier of the area in which the node is located. The entire packet, except the authentication data, is protected by a 16-bit checksum using the same algorithm as the IP header .The Authentication type is 0 if no authentication is used; otherwise, it may be 1, implying that a simple password is used, or 2, which indicates that a cryptographic authentication checksum is used. In the 2nd cases, the Authentication field carries the password or cryptographic checksum.



Of the five OSPF message types, type 1 is the —hello‖ message, which a router sends to its peers to notify them that it is still alive and connected. The remaining types are used to request, send, and acknowledge the receipt of link-state messages.

The basic building block of link-state messages in OSPF is the link-state advertisement (LSA). One message may contain many LSAs. Specifically, a router running OSPF may generate link-state packets that advertise one or more of the networks that are directly connected to that router. In addition, a router that is connected to another router by some link must advertise the cost of reaching that router over the link.



The figure below shows the packet format for a type 1 link-state advertisement. Type 1 LSAs advertise the cost of links between routers. Type 2 LSAs are used to advertise networks to which the advertising router is connected, while other types are used to support additional hierarchy.

- The LS Age is the equivalent of a time to live. The Type field tells us that this is a type 1 LSA.
- The LS sequence number is used exactly to detect old or duplicate LSAs. The LS checksum is used to verify that data has not been corrupted. It covers all fields in the packet except LS Age.
- Length is the length in bytes of the complete LSA.
- The first two of these fields identify the link; The metric is of course the cost of the link. Type tells us something about the link—for example, if it is a point-to-point link.
- The TOS information is present to allow OSPF to choose different routes for IP packets based on the value in their TOS field.

---

2. **Write about IPv6 in detail. What are its new features and improvements. [NOV/DEC 10,11], [MAY/JUNE 2012],[MAY/JUNE-2016]**

Larger address space. An IPv6 address is 128 bits long, 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.

New options. IPv6 has new options to allow for additional functionalities.

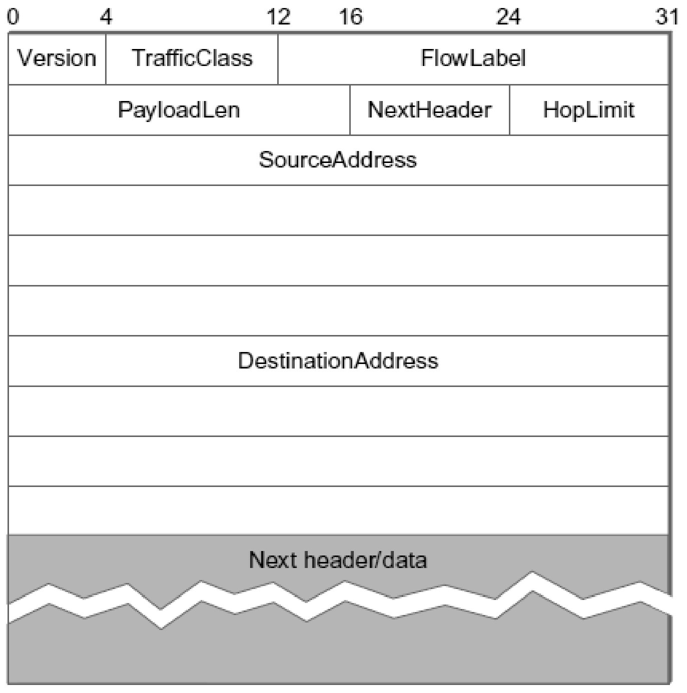
Allowance for extension. IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.

Support for resource allocation. In IPv6, the type-of-service field has been removed, but a mechanism (called flow *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 Version field is in the same place relative to the start of the header as IPv4's Version field so that header-processing software can immediately decide which header format to look for.
- The PayloadLen field gives the length of the packet, excluding the IPv6 header, measured in bytes.
- The Next Header field cleverly replaces both the IP options and the Protocol field of IPv4.
- The Hop Limit field is simply the TTL of IPv4,
- The IPv6 header is always 40 bytes long.
- The longer address format in IPv6 helps provide a useful, new form of auto configuration called *stateless* auto configuration, which does not require a server.



3. What is internet Multicasting? Explain in detail. [NOV/DEC-14,15,April 17] Explain in detail Multicast routing (DVMRP, PIM) [Regulation 2013]

Multicast Routing (DVMRP)

Multicast routing is the process by which the multicast distribution trees are determined or, more concretely, the process by which the multicast forwarding tables are built.

DVMRP

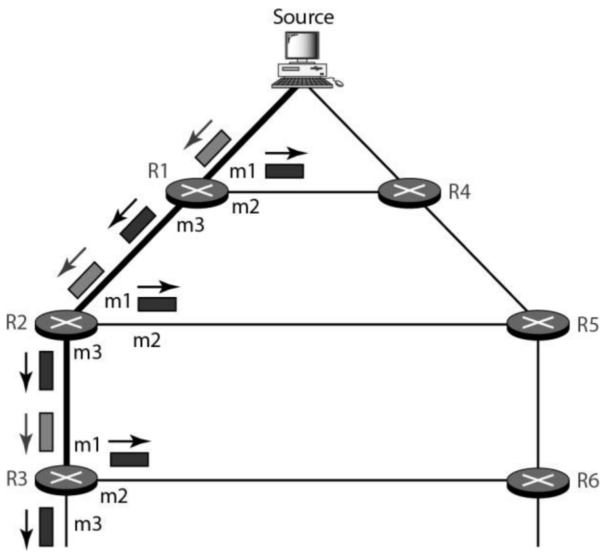
Multicast routing does not allow a router to send its routing table to its neighbors..Multicast distance vector routing uses source-based trees, but the router never actually makes a routing table. When a router receives a multicast packet, it forwards the packet as though it is consulting a routing table. The multicast distance vector algorithm uses a process based on four decision-making strategies. Each strategy is built on its predecessor.

Flooding.

A router receives a packet and, without even looking at the destination group address, sends it out from every interface 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.

The next strategy, reverse path forwarding, corrects this defect.

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.

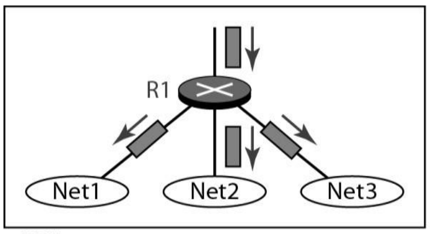


The shortest path tree as calculated by routers RI, R2, and R3 is shown by a thick line. When RI receives a packet from the source through the interface rnl, it consults its routing table and finds that the shortest path from RI to the source is through interface mI. The packet is forwarded

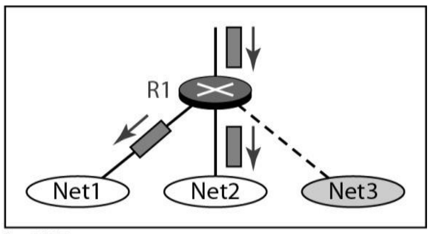
Reverse Path Broadcasting (RPB)

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.

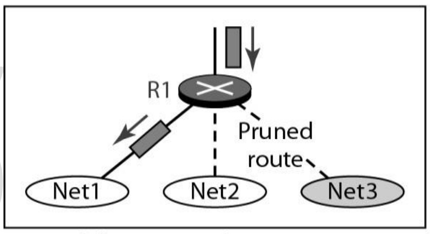
Reverse Path Multicasting (RPM).



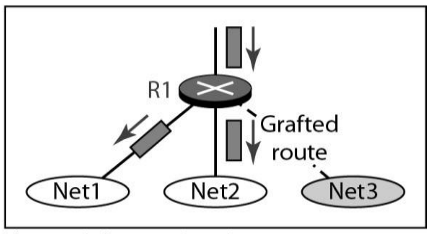
a.RPF



b. RPB



c. RPM (after pruning)



d. RPM (after grafting)

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)

PIM-SM

Protocol Independent Multicast, or PIM, was developed in response to the scaling problems of earlier multicast routing protocols.

PIM dense mode (PIM-DM) uses a flood-and-prune algorithm like DVMRP and suffers from the same scalability problem. PIM sparse mode (PIM-SM) has become the dominant multicast routing protocol.

In PIM-SM, routers explicitly join the multicast distribution tree using PIM protocol messages known as Join messages. The question that arises is where to send those Join messages because, after all, any host (and any number of hosts) could send to the multicast group. To address this, PIM-SM assigns to each group a special router known as the *rendezvous point* (RP).

PIM-SM defines a set of procedures by which all the routers in a domain can agree on the router to use as the RP for a given group.

A multicast forwarding tree is built as a result of routers sending Join messages to the RP. PIM-SM allows two types of trees to be constructed: a *shared* tree, which may be used by all senders, and a *source-specific* tree, which may be used only by a specific sending host.
In the figure below (a), in which router R4 is sending a Join to the rendezvous point for some group.

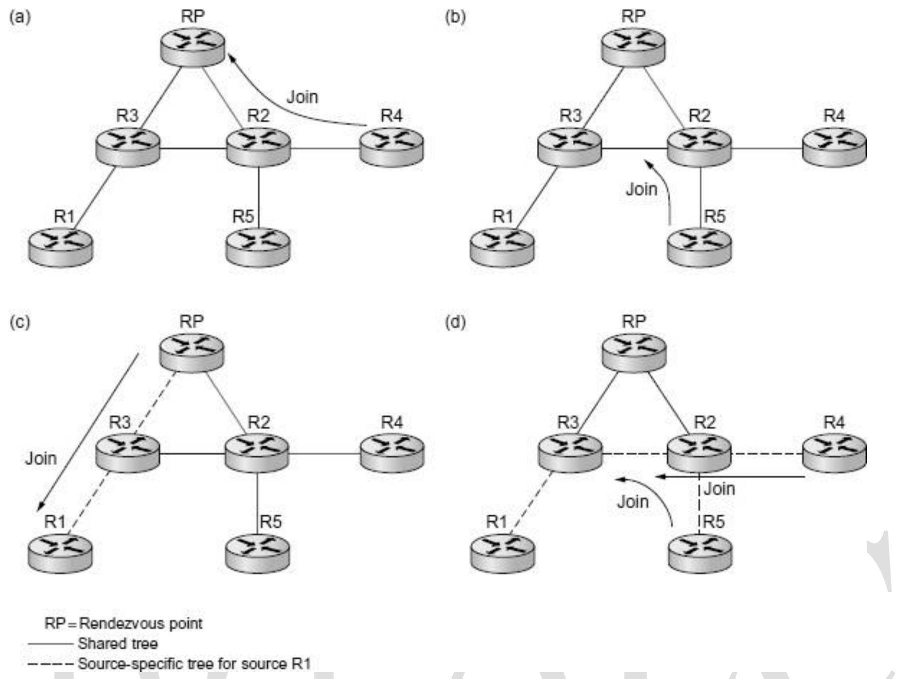


Fig:. PIM operation: (a) R4 sends Join to RP and joins shared tree; (b) R5 joins shared tree; (c) RP builds source-specific tree to R1 by sending Join to R1; (d) R4 and R5 build source-specific tree to R1 by sending Joins to R1.

A Join message clearly must pass through some sequence of routers before reaching the RP (e.g., R2). Each router along the path looks at the Join and creates a forwarding table entry for the shared tree, called a (*, G) entry (where * means ―all senders‖).

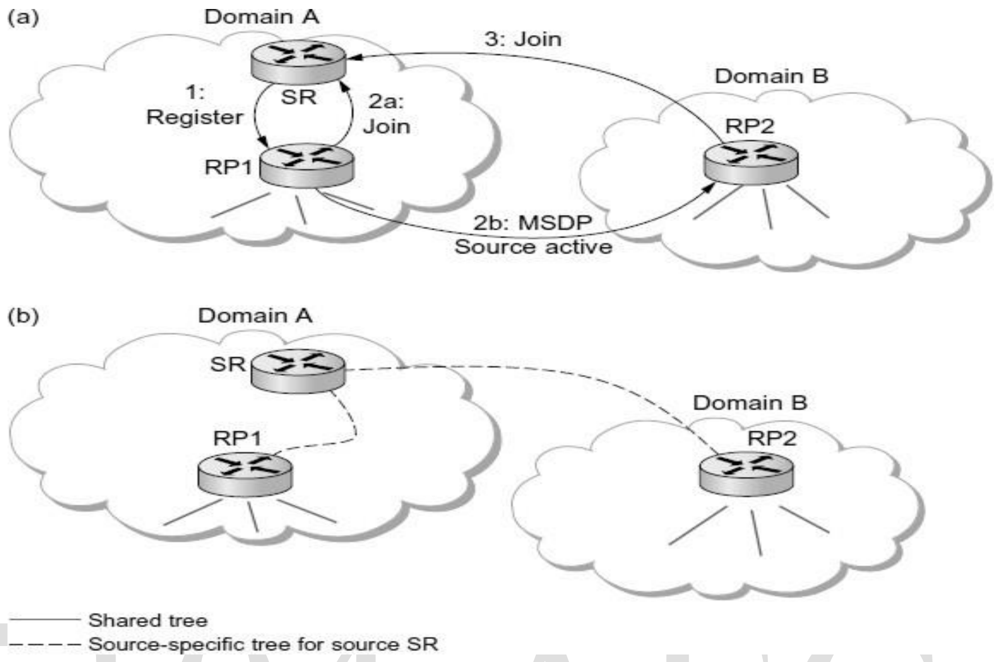
To create the forwarding table entry, it looks at the interface on which the Join arrived and marks that interface as one on which it should forward data packets for this group.

Source-Specific Multicast (PIM-SSM)

The newly exposed capability is now known as PIM-SSM (PIM Source-Specific Multicast). PIM-SSM introduces a new concept, the *channel*, which is the combination of a source address S and a group address G.

To use PIM-SSM, a host specifies both the group and the source in an IGMP Membership Report message to its local router.

That router then sends a PIM-SM source-specific Join message toward the source, thereby adding a branch to itself in the source-specific tree, just as was described above for ―normal‖ PIM-SM, but bypassing the whole shared-tree stage. Since the tree that results is source specific, only the designated source can send packets on that tree.



The introduction of PIM-SSM has provided some significant benefits, particularly since there is relatively high demand for one-to-many multicasting:

- Multicasts travel more directly to receivers.
- The address of a channel is effectively a multicast group address plus a source address. Therefore, given that a certain range of multicast group addresses will be used for SSM exclusively, multiple domains can use the same multicast group address independently and without conflict, as long as they use it only with sources in their own domains.
- Because only the specified source can send to an SSM group, there is less risk of attacks based on malicious hosts overwhelming the routers or receivers with bogus multicast traffic.
- PIM-SSM can be used across domains exactly as it is used within a domain, without reliance on anything like MSDP. SSM, therefore, is quite a useful addition to the multicast service model.

4. Describe the Interdomain routing (Border Gateway Protocol).

Challenges in Interdomain Routing Perhaps the most important challenge of interdomain routing today is the need for each AS to determine its own routing *policies*.

The Border Gateway Protocol (BGP), which is in its fourth version at the time of this writing (BGP-4). BGP is often regarded as one of the more complex parts of the Internet.

We can classify autonomous systems into three broad types:

- **Stub AS**—an AS that has only a single connection to one other AS; such an AS will only carry local traffic. The small corporation in the figure below is an example of a stub AS.
- **Multihomed AS**—an AS that has connections to more than one other AS but that refuses to carry transit traffic, such as the large corporation at the top of the figure below.
- **Transit AS**—an AS that has connections to more than one other AS and that is designed to carry both transit and local traffic,

First, it is necessary to find *some* path to the intended destination that is loop free. Second, paths must be compliant with the policies of the various autonomous systems along the path

Basics of BGP

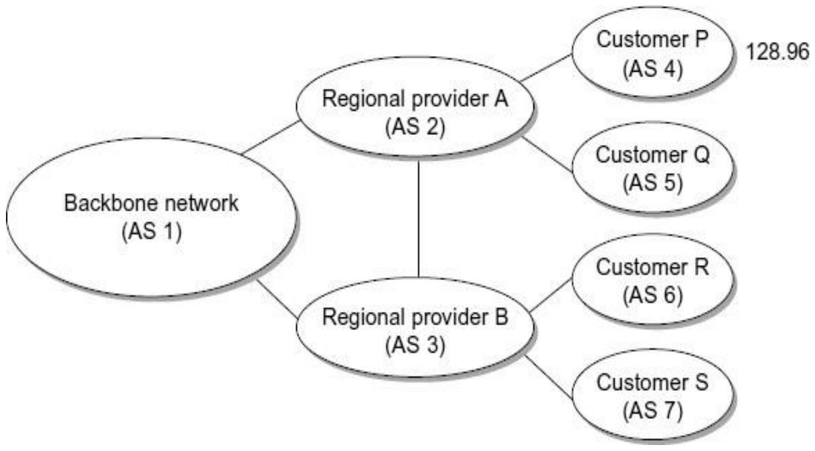
Each AS has one or more *border routers* through which packets enter and leave the AS. A border router is simply an IP router that is charged with the task of forwarding packets between autonomous systems. Each AS that participates in BGP must also have at least one *BGP* speaker, a router that ―speaks‖ BGP to other BGP speakers in other autonomous systems.

BGP advertises *complete paths* as an enumerated list of autonomous systems to reach a particular network.

It also enables routing loops to be readily detected. Consider the very simple example network in the figure below. Assume that the providers are transit networks, while the customer networks are stubs.

A BGP speaker for the AS of provider A (AS 2) would be able to advertise reachability information for each of the network numbers assigned to customers P and Q. The networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached directly from AS 2.‖

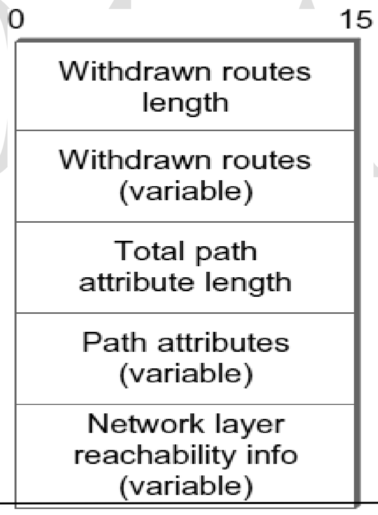
The backbone network, on receiving this advertisement, can advertise, ―The networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS 1, AS 2).‖ Similarly, it could advertise, ―The networks 192.12.69, 192.4.54, and 192.4.23 can be reached along the path (AS 1, AS 3).‖ An important job of BGP is to prevent the establishment of looping paths.



A given AS will only advertise routes that it considers good enough for itself.

BGP speakers need to be able to cancel previously advertised paths. This is done with a form of negative advertisement known as a *withdrawn route*.

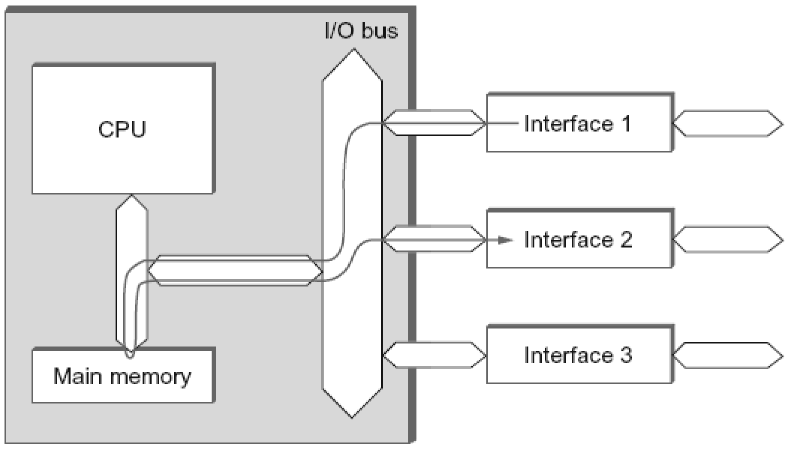
Both positive and negative reachability information are carried in a BGP update message, the format of which is shown in the figure below.A BGP speaker can simply send an occasional *keepalive* message that says, in effect, –I'm still here and nothing has changed.‖.


0
15
Withdrawn routes length
Withdrawn routes (variable)
Total path attribute length
Path attributes (variable)
Network layer reachability info (variable)

5. Explain in detail about Switch basics and Routing Areas [Regulation 2013]. Switch Basics:

Switches and routers use similar implementation techniques, the figure below shows a processor with three network interfaces used as a switch. The figure shows a path that a packet might take from the time it arrives on interface 1 until it is output on interface 2. It is assumed here that the processor has a mechanism to move data directly from an interface to its main memory without having to be directly copied by the CPU, a technique called *direct memory access* (DMA).

Once the packet is in memory, the CPU examines its header to determine which interface the packet should be sent out on. It then uses DMA to move the packet out to the appropriate interface. The figure above does not show the packet going to the CPU because the CPU inspects only the header of the packet; it does not have to read every byte of data in the packet.



The main problem with using a general-purpose processor as a switch is that its performance is limited by the fact that all packets must pass through a single point of contention: In the example shown, each packet crosses the I/O bus twice and is written to and read from main memory once.

The upper bound on aggregate throughput of such a device is, thus, either half the main memory bandwidth or half the I/O bus bandwidth, whichever is less. Moreover, this upper bound also assumes that moving data is the only problem—a fair approximation for long packets but a bad one when packets are short. In the latter case, the cost of processing each packet— parsing its header and deciding which output link to transmit it on—is likely to dominate.

Suppose, for example, that a processor can perform all the necessary processing to switch 2 million packets each second. This is sometimes called the packet per second (pps) rate. If the average packet is short, say, 64 bytes, this would imply that is, a throughput of about 1 Gbps—substantially below the range that users are demanding from their networks today.

$$\text{Throughput} = \text{pps}\times(\text{BitsPerPacket})$$

$$= 2\times106\times64\times8$$

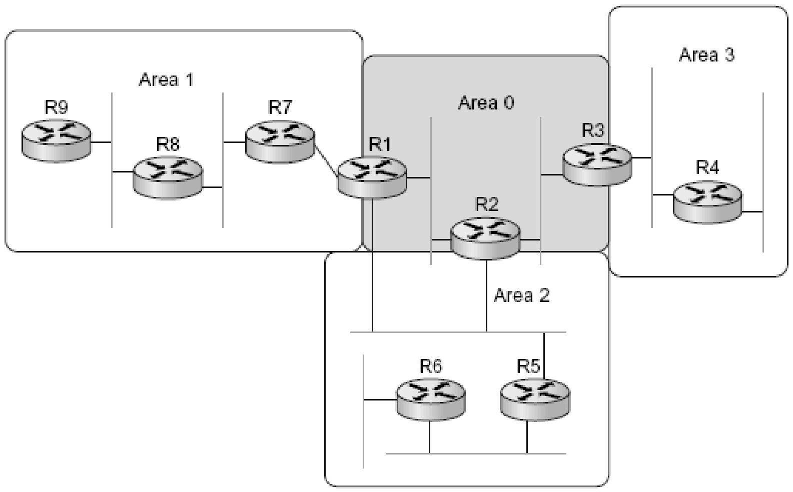
$$= 1024\times106$$

Routing Areas

Link-state routing protocols (such as OSPF and IS-IS) can be used to partition a routing domain into subdomains called *areas*.

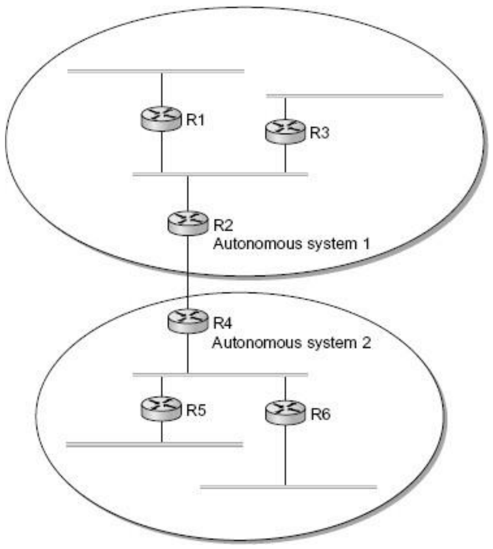
An area is a set of routers that are administratively configured to exchange link-state information with each other. There is one special area—the backbone area, also known as area 0. An example of a routing domain divided into areas is shown in the figure below.

Routers R1, R2, and R3 are members of the backbone area. They are also members of at least one nonbackbone area; R1 is actually a member of both area 1 and area 2. A router that is a member of both the backbone area and a nonbackbone area is an area border router (ABR).



All the routers in the area send link-state advertisements to each other and thus develop a complete, consistent map of the area. However, the link-state advertisements of routers that are not area border routers do not leave the area in which they originated. For example, router R4 in area 3 will never see a link-state advertisement from router R8 in area 1. As a consequence, it will know nothing about the detailed topology of areas other than its own

To make this work, the area border routers summarize routing information that they have learned from one area and make it available in their advertisements to other areas



The use of areas forces all packets travelling from one area to another to go via the backbone area, even if a shorter path might have been available. For example, even if R4 and R5 were directly connected, packets would not flow between them because they are in different nonbackbone areas. It turns out that the need for scalability is often more important than the need to use the absolute shortest path.

PART B

1. Explain in detail about TCP congestion avoidance algorithms.

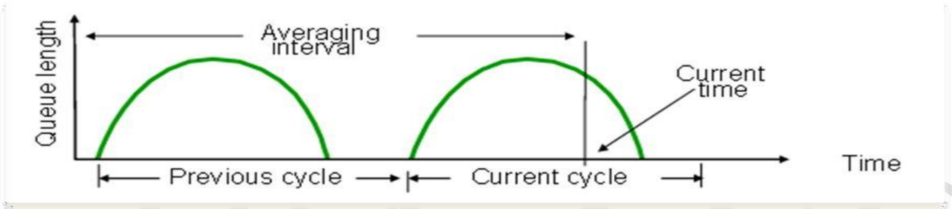
[NOV/DEC 2011] , [MAY/JUNE2014]

Congestion avoidance mechanisms prevent congestion before it actually occurs. TCP creates loss of packets in order to determine bandwidth of the connection. Routers help the end nodes by intimating when congestion is likely to occur. Congestion-avoidance mechanisms are:

- DEC bit and Random Early Detection (RED)

DEC bit is a technique implemented in routers to avoid congestion. Its utility is to predict possible congestion and prevent it. This protocol works with TCP.

When a router wants to signal congestion to the sender, it adds a bit in the header of packets sent. When a packet arrives at the router, the router calculates the average queue length for the last (busy + idle) period plus the current busy period.. When the average queue length exceeds 1, then the router sets the congestion indication bit in the packet header of arriving packets.



When the destination replies, the corresponding ACK includes a bit of congestion. The sender receives the ACK and calculates how many packets it received with the congestion indication bit set to one. If less than half of the packets in the last window had the congestion indication bit set, then the window is increased linearly. Otherwise, the window is decreased exponentially.

This technique provides distinct advantages:

- Dynamically manages the window to avoid congestion and increasing freight if it detects congestion.
- Try to balance bandwidth with respect to the delay.

The Dec Bit mechanism, also known as the **Congestion Indication mechanism**, is a *binary feedback* congestion avoidance mechanism developed for the Digital Network Architecture at DEC , and has since been specified as the congestion avoidance mechanism for the ISO TP4 and CLNP transport and network protocols.

In Dec Bit, all network packets have a single bit, the `Congestion Experienced Bit', in their headers. Sources set this bit to zero. If the packet passes through a router that believes itself to be congested, it sets the bit to a one.

Acknowledgment packets from the destination return the received Congestion Experienced Bit to the source. If the bit is set, the source knows there was some congestion along the path to the destination, and takes remedial action.

In DECNET, the source adjusts its window size. In TP4, the destination alters the advertised window size, rather than returning the bit to the source.

Dec Bit can be used as either congestion avoidance or a congestion recovery mechanism, but is used as an avoidance mechanism .To be used as an avoidance mechanism, routers must determine if their load is at, below or above their peak power point.

This is achieved by averaging the size of the router's output queues. If they have more than one buffered packet on average, that output interface's load is above the peak power point; if less than one packet, the interface's load is below the peak power point.

Determining a good averaging function is not easy. Using instantaneous queue sizes is an inaccurate indication of the load on an interface. On the other hand, a too-large averaging interval gives a distorted indication of the real load. They settled for an average taken from the second-last `regeneration point' to the current time, where a `regeneration point' is a point when the queue size is zero.

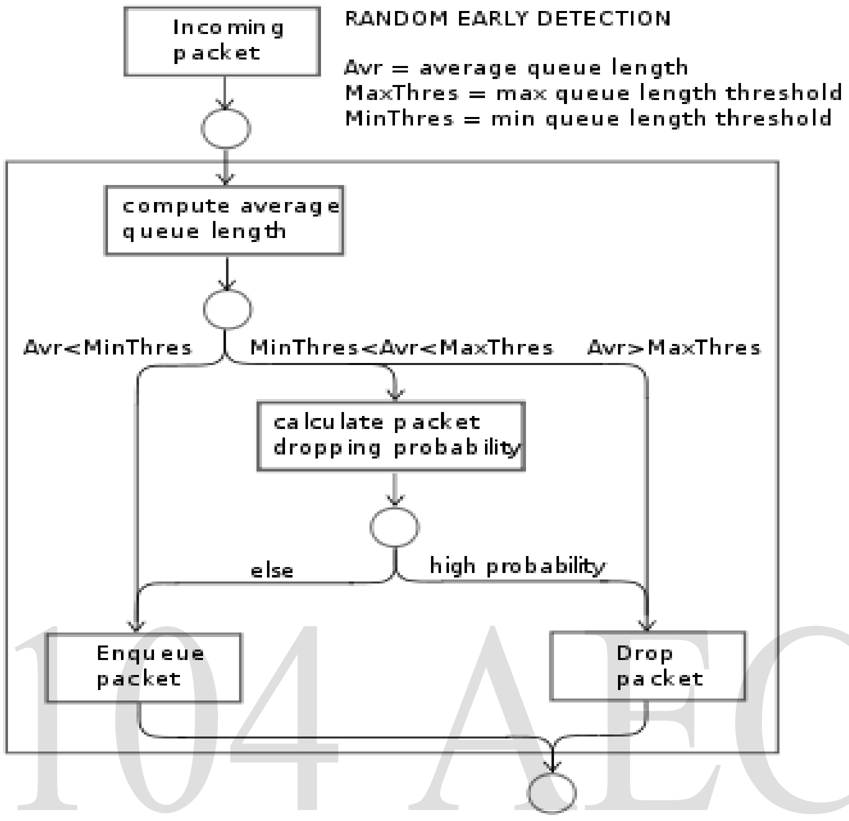
RED (Random Early Detection)

Random early detection (**RED**), also known as **random early discard** or **random early drop** is a queuing discipline for a network scheduler suited for congestion avoidance.

In the conventional tail drop algorithm, a router or other network component buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows.

Tail drop can also lead to TCP global synchronization as all TCP connections "hold back" simultaneously, and then step forward simultaneously. Networks become under-utilized and flooded by turns. RED addresses these issues.

RED monitors the average queue size and drops packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped.


Incoming packet
RANDOM EARLY DETECTION
Avr = average queue length
MaxThres = max queue length threshold
MinThres = min queue length threshold
compute average queue length
Avr<MinThres
MinThres<Avr<MaxThres
Avr>MaxThres
calculate packet dropping probability
else
high probability
Enqueue packet
Drop packet

A RED router randomly drops incoming packets when it believes that is is becoming congested, implicitly notifying the source of network congestion by the packet loss. RED is essentially designed to work with TCP.

Random Early Drop is reasonably simple, appropriate for a wide range of environments, and does not require modifications to existing TCP implementations. Random Early Drop does not work with non-TCP protocols, still causes packet loss, and still takes time to find equilibrium..

2. Explain TCP congestion control mechanisms in detail.

[NOV/DEC 16,12,MAY/JUNE2014,2016,15,17]

Congestion control mechanisms are:

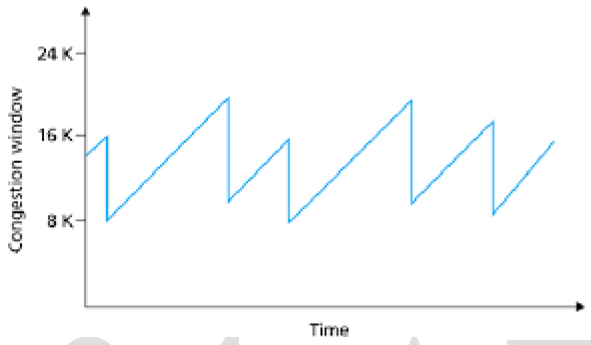
- Additive Increase / Multiplicative Decrease (AIMD)
- Slow Start
- Fast Retransmit and Fast Recover

The additive-increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm best known for its use in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. Multiple flows using AIMD congestion control will eventually converge to use equal amounts of a contended link.

Algorithm

The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs..

When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half after loss. The result is a saw-tooth behavior that represents the probe for bandwidth.



AIMD requires a binary signal of congestion. Most frequently, packet loss serves as the signal; the multiplicative decrease is triggered when a timeout or acknowledgement message indicates a packet was lost.

Mathematical Formula

Let *w*(*t*) be the sending rate (e.g. the congestion window) during time slot *t*, *a* ($a > 0$) be the additive increase parameter, and *b* ($0 < b < 1$) be the multiplicative decrease factor.

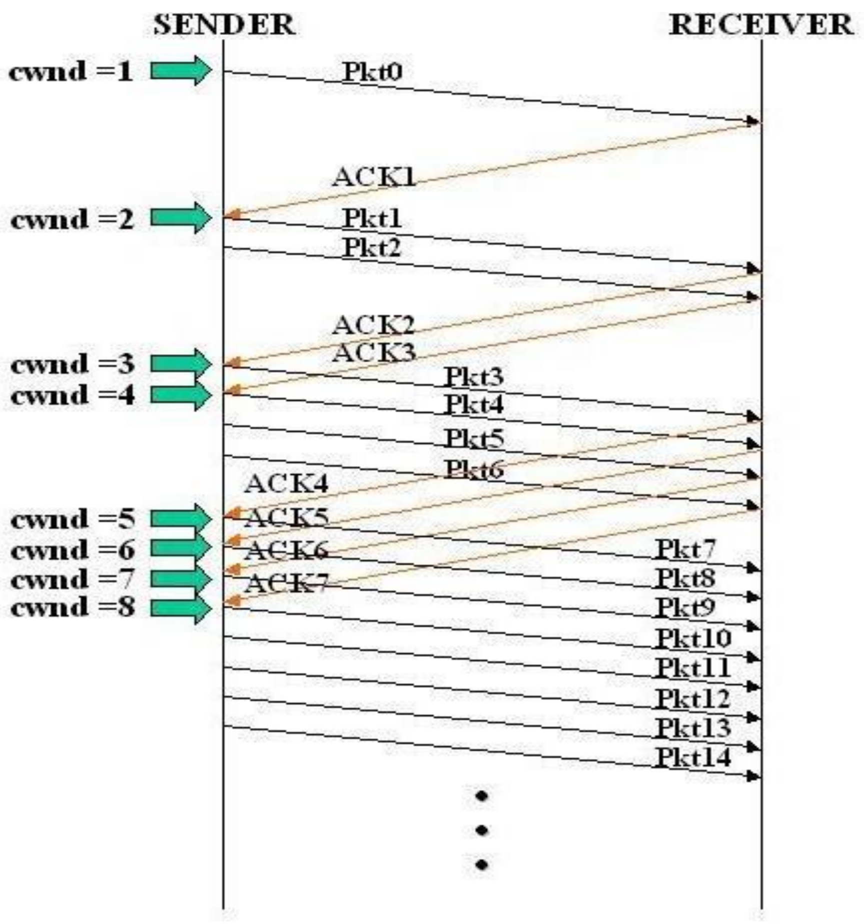
$$w(t+1) = \begin{cases} w(t) + a & \text{if congestion is not detected} \\ w(t) \times b & \text{if congestion is detected} \end{cases}$$

A congestion-control strategy that only decreases the window size is obviously too conservative. We also need to be able to increase the congestion window to take advantage of newly available capacity in the network. This is the ―additive increase‖ part of AIMD, and it works as follows. Every time the source successfully sends a Congestion Window‘s worth of packets—that is, each packet sent out during the last round-trip time (RTT) has been ACKed—it adds the equivalent of 1 packet to Congestion Window

Slow-start

- Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion.

- Slow-start is one of the algorithms that TCP uses to control congestion inside the network. It is also known as the exponential growth phase.
- Slow-start begins initially with a congestion window Size (cwnd) of 1, 2 or 10. The value of the Congestion Window will be increased with each acknowledgment received, effectively doubling the window size each round trip .
- The transmission rate will be increased with slow-start algorithm until either a loss is detected, or the receiver's advertised window (rwnd) is the limiting factor, or the slow start threshold (ssthresh) is reached. If a loss event occurs, TCP assumes that it is due to network congestion and takes steps to reduce the offered load on the network
- Once ssthresh is reached, TCP changes from slow-start algorithm to the linear growth (congestion avoidance) algorithm. At this point, the window is increased by 1 segment for each RTT.

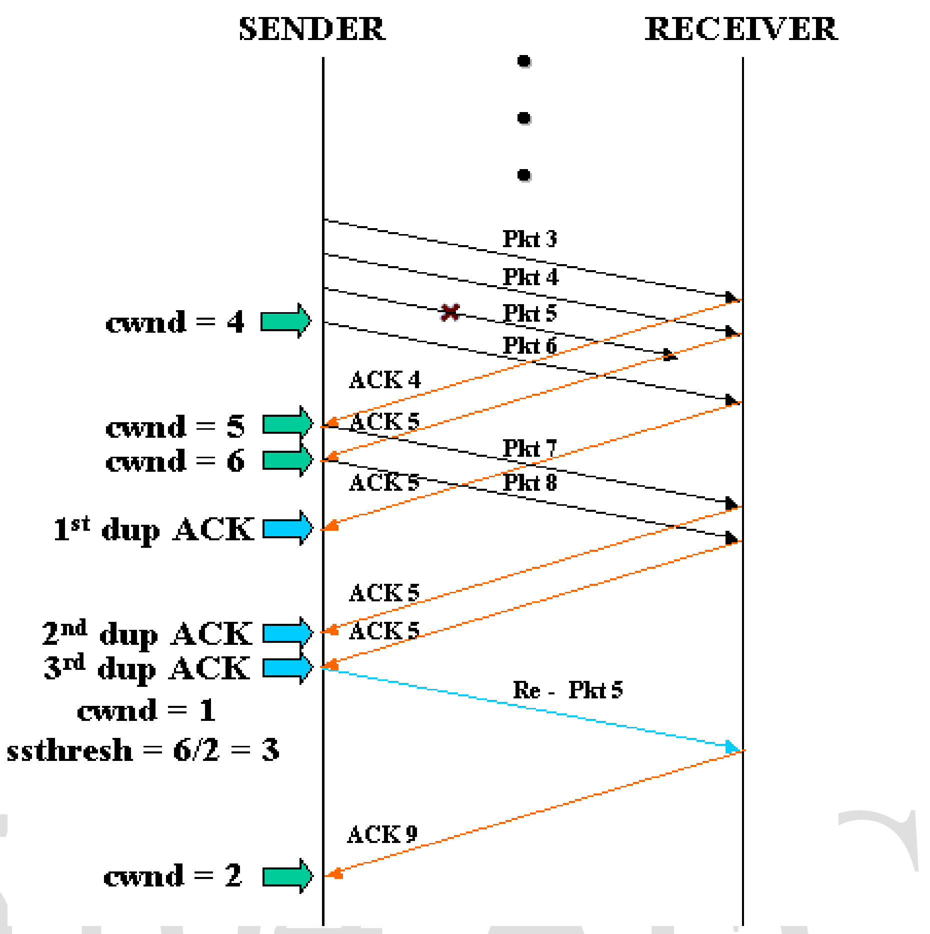


.

Although the strategy is referred to as "Slow-Start", its congestion window growth is quite aggressive, more aggressive than the congestion avoidance phase. Before slow-start was introduced in TCP, the initial pre-congestion avoidance phase was even faster.

Fast Retransmit

- Fast retransmit is an enhancement to TCP which reduces the time a sender waits before retransmitting a lost segment.
- A TCP sender uses a timer to recognize lost segments. If an acknowledgement is not received for a particular segment within a specified time (a function of the estimated Round-trip delay time), the sender will assume the segment was lost in the network, and will retransmit the segment.
- Duplicate acknowledgement is the basis for the fast retransmit mechanism which works as follows: after receiving a packet (e.g. with sequence number 1), the receiver sends an acknowledgement by adding 1 to the sequence number (i.e., acknowledgement number 2) which means that the receiver receives the packet number 1 and it expects packet number 2 from the sender.
- Let's assume that three subsequent packets have been lost. In the meantime the receiver receives packet numbers 5 and 6. After receiving packet number 5, the receiver sends an acknowledgement, but still only for sequence number 2. When the receiver receives packet number 6, it sends yet another acknowledgement value of 2. Because the sender receives more than one acknowledgement with the same sequence number (2 in this example) this is called *duplicate acknowledgement*.



The fast retransmit enhancement works as follows: if a TCP sender receives a specified number of acknowledgements which is usually set to three duplicate acknowledgements with the same acknowledge number, the sender can be reasonably confident that the segment with the next higher sequence number was dropped, and will not arrive out of order. The sender will then retransmit the packet that was presumed dropped before waiting for its timeout.

The timer is usually set to the *retransmission timeout period (RTO)* which is determined by some other algorithms. If TCP correctly receives an ACK corresponding to the data packet before the timer is expired. TCP, then, automatically resets the timer of just received ACK packet and continuously waits for the other ACK packets.

Fast retransmit is a heuristic that sometimes triggers the retransmission of a dropped packet before the RTO period is up. The timeout mechanism actives normally for a small window size, where packets in transit are not enough to cause fast retransmit. TCP can employ fast retransmit only in a large window size to enhance its performance and link utilization.

The idea of fast retransmit is straightforward. It only adds a tiny thing to the normal operation of TCP. Every time a packet with sequence number x arrives correctly at the receiver; the receiver acknowledges the packet #x by sending an ACK packet (containing the sequence number of another packet which it is waiting for - this number may or may not be "x+1") back to the sender.

Fast retransmit plays an important role here. After receiving some numbers of duplicate ACKs, TCP at the sending side retransmits the missing packet without waiting for the timer to be expired. Moreover, receiving some numbers of duplicate ACKs means that the network congestion has been occurred.

Thus, TCP at the sending side resets cwnd to 1 and sets ssthresh to (old cwnd / 2) due to the congestion control algorithm; then starts slow-start again. In the practical TCP, the third duplicate ACKs triggers fast retransmit.

---

3. Draw and explain TCP state transition diagram [NOV/DEC-2015]

With neat Architecture, Explain TCP in detail [MAY/JUNE 2013]

TCP is a **connection-oriented** protocol; a formal relationship (handshake) is established before exchanging data. The system that initiates the connection is considered as the **client** in the TCP terminology while the system that accepts this connection is considered as the **server**.

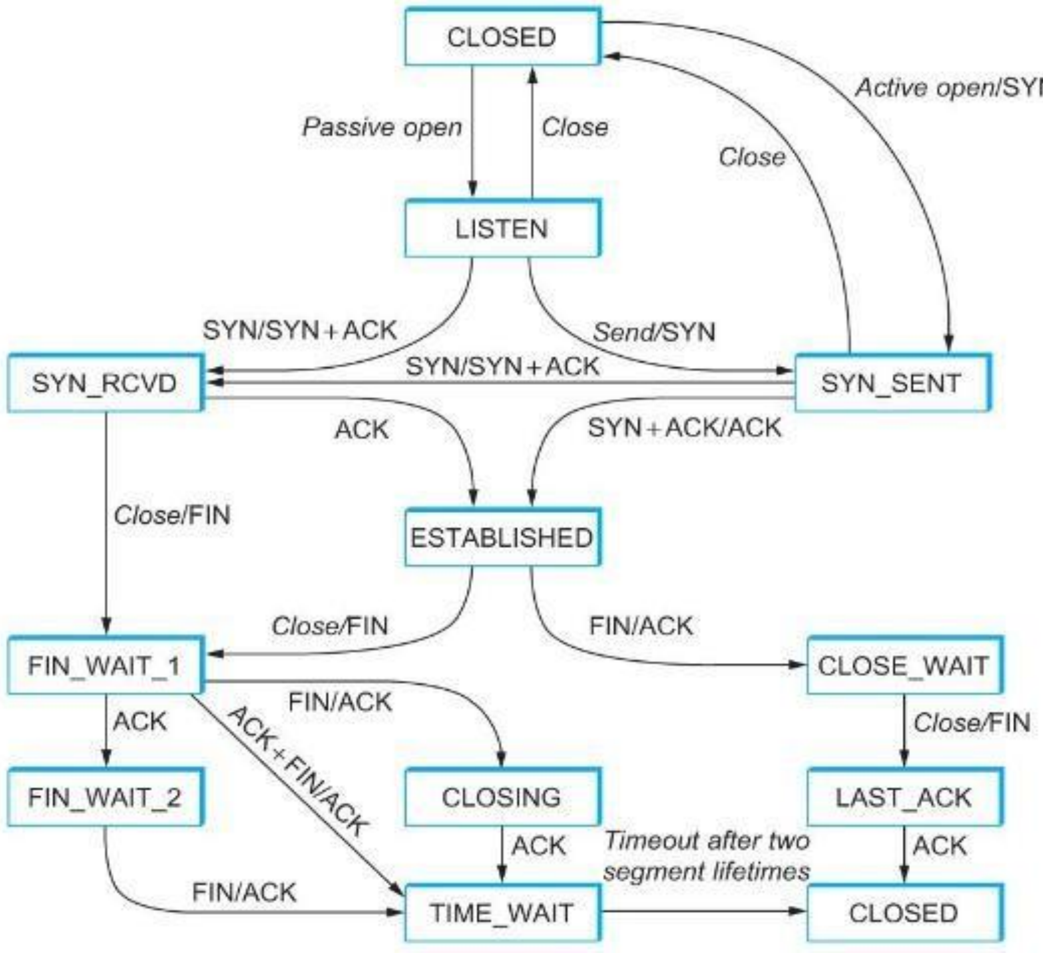
Two systems can establish connections to one another and simultaneously, in this case they are both server and client. The client and server exchange units of information called "**TCP segments**, the segments being composed of a header and a data area.

States involved in opening and closing a connection is shown above and below ESTABLISHED state respectively. Operation of sliding window is hidden in the ESTABLISHED state events that trigger a state transition is:

- Segments that arrive from its peer.
- Application process invokes an operation on TCP

TCP is complex enough that its specification includes a state-transition diagram. A copy of this diagram is given in the figure below. This diagram shows only the states involved in opening a connection (everything above ESTABLISHED) and in closing a connection (everything below ESTABLISHED). Everything that goes on while a connection is open—that is, the operation of the sliding window algorithm—is hidden in the ESTABLISHED state. TCP‘s state-transition diagram is fairly easy to understand. Each circle denotes a state that one end of a TCP connection can find itself in. All connections start in the CLOSED state.

As the connection progresses, the connection moves from state to state according to the arcs. Each arc is labeled with a tag of the form *event/action*. Thus, if a connection is in the LISTEN state and a SYN segment arrives (i.e., a segment with the SYN flag set), the connection makes a transition to the SYN RCVD state and takes the action of replying with an ACK+SYN segment.



Notice that two kinds of events trigger a state transition:
(1) A segment arrives from the peer (e.g., the event on the arc from LISTEN to SYNRCVD)
(2) The local application process invokes an operation on TCP (e.g., the *active open* event on the arc from CLOSED to SYN SENT). In other words, TCP‘s state-transition diagram effectively defines the *semantics* of both its peer-to-peer interface and its service interface. The *syntax* of these two interfaces is given by the segment format and by some application programming interface respectively.

TCP Finite State Machine (FSM) States, Events and Transitions		
State	**State Description**	**Event and Transition**
CLOSED	This is the default state that each connection starts in before the process of establishing it begins. The state is called ―fictional‖ in the standard. The reason is that this state represents the situation where there is no connection between devices—it either hasn't	**Passive Open:** A server begins the process of connection setup by doing a passive open on a TCP port. At the same time, it sets up the data structure (transmission control block or TCB) needed to manage the connection. It then transitions to the *LISTEN* state.

	been created yet, or has just been destroyed. If that makes sense.	**Active Open, Send *SYN*:** A client begins connection setup by sending a *SYN* message, and also sets up a TCB for this connection. It then transitions to the *SYN-SENT* state.
LISTEN	A device (normally a server) is waiting to receive a *synchronize* (*SYN*) message from a client. It has not yet sent its own *SYN* message.	**Receive Client *SYN*, Send *SYN+ACK*:** The server device receives a *SYN* from a client. It sends back a message that contains its own *SYN* and also acknowledges the one it received. The server moves to the *SYN-RECEIVED* state.
SYN-SENT	The device (normally a client) has sent a *synchronize* (*SYN*) message and is waiting for a matching *SYN* from the other device (usually a server).	**Receive *SYN*, Send *ACK*:** If the device that has sent its *SYN* message receives a *SYN* from the other device but not an *ACK* for its own *SYN*, it acknowledges the *SYN* it receives and then transitions to *SYN-RECEIVED* to wait for the acknowledgment to its *SYN*.
		Receive *SYN+ACK*, Send *ACK*: If the device that sent the *SYN* receives both an acknowledgment to its *SYN* and also a *SYN* from the other device, it acknowledges the *SYN* received and then moves straight to the *ESTABLISHED* state.
SYN-RECEIVED	The device has both received a *SYN* (connection request) from its partner and sent its own *SYN*. It is now waiting for an *ACK* to its *SYN* to finish connection setup.	**Receive *ACK*:** When the device receives the *ACK* to the *SYN* it sent, it transitions to the *ESTABLISHED* state.
ESTABLISHED	The ―steady state‖ of an open TCP connection. Data can be exchanged freely once both devices in the connection enter this state. This will continue until the connection is closed for one reason or another.	**Close, Send *FIN*:** A device can close the connection by sending a message with the *FIN* (*finish*) bit sent and transition to the *FIN-WAIT-1* state.
		Receive *FIN*: A device may receive a *FIN* message from its connection partner asking that the connection be

		closed. It will acknowledge this message and transition to the *CLOSE-WAIT* state.
CLOSE-WAIT	The device has received a close request (*FIN*) from the other device. It must now wait for the application on the local device to acknowledge this request and generate a matching request.	**Close, Send *FIN*:** The application using TCP, having been informed the other process wants to shut down, sends a close request to the TCP layer on the machine upon which it is running. TCP then sends a *FIN* to the remote device that already asked to terminate the connection. This device now transitions to *LAST-ACK*.
LAST-ACK	A device that has already received a close request and acknowledged it, has sent its own *FIN* and is waiting for an *ACK* to this request.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. We have now sent our *FIN* and had it acknowledged, and received the other device's *FIN* and acknowledged it, so we go straight to the *CLOSED* state.
FIN-WAIT-1	A device in this state is waiting for an *ACK* for a *FIN* it has sent, or is waiting for a connection termination request from the other device.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. It transitions to the *FIN-WAIT-2* state.
		Receive *FIN*, Send *ACK*: The device does not receive an *ACK* for its own *FIN*, but receives a *FIN* from the other device. It acknowledges it, and moves to the *CLOSING* state.
FIN-WAIT-2	A device in this state has received an ACK for its request to terminate the connection and is now waiting for a matching *FIN* from the other device.	**Receive *FIN*, Send *ACK*:** The device receives a *FIN* from the other device. It acknowledges it and moves to the *TIME-WAIT* state.
CLOSING	The device has received a *FIN* from the other device and sent an *ACK* for it, but not yet received an *ACK* for its own *FIN* message.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. It transitions to the *TIME-WAIT* state.

TIME-WAIT	The device has now received a *FIN* from the other device and acknowledged it, and sent its own *FIN* and received an *ACK* for it. We are done, except for waiting to ensure the ACK is received and prevent potential overlap with new connections.	**Timer Expiration:** After a designated wait period, device transitions to the *CLOSED* state.

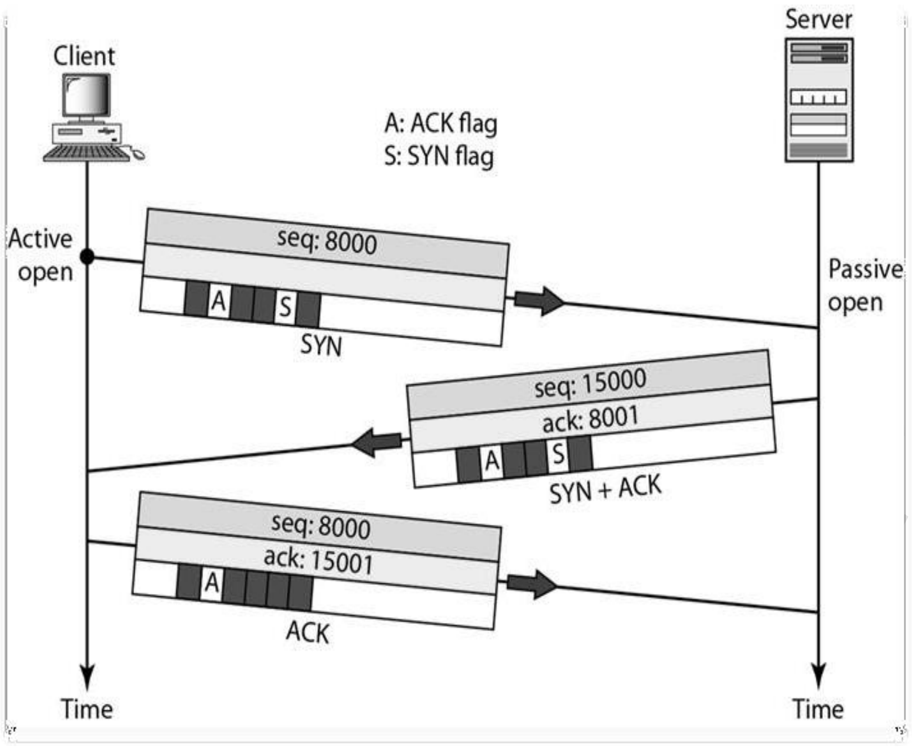
4. Explain TCP 3 way Handshake in detail?[APR/MAY-2015] TCP 3-Way Handshake

The TCP connection is set up via three-way handshaking:

- This begins with a **SYN** (Synchronize) segment (as indicated by the code bit) containing a 32-bit **Sequence** number **A** called the **Initial Send Sequence (ISS)** being chosen by, and sent from, host 1. This 32-bit sequence number **A** is the starting sequence number of the data in that packet and increments by **1** for every byte of data sent within the segment, i.e. there is a sequence number for each octet sent. The SYN segment also puts the value **A+1** in the first octet of the data.
- Host 2 receives the **SYN** with the Sequence number **A** and sends a **SYN** segment with its own totally independent ISS number **B** in the Sequence number field. In addition, it sends an increment on the Sequence number of the last received segment (i.e. **A+x** where **x** is the number of octets that make up the data in this segment) in its Acknowledgment field. This **Acknowledgment** number informs the recipient that its data was received at the other end and it expects the next segment of data bytes to be sent, to start at sequence number **A+x**. This stage is aften called the **SYN-ACK**. It is here that the **MSS** is agreed.
- Host 1 receives this **SYN-ACK** segment and sends an **ACK** segment containing the next sequence number (**B+y** where **y** is the number of octets in this particular segment), this is called **Forward Acknowledgement** and is received by Host 2. Segments that are not acknowledged within a certain time span are retransmitted.

TCP peers must not only keep track of their own initiated Sequence numbers but also those Acknowledgment numbers of their peers.

Closing a TCP connection is achieved by the initiator sending a **FIN** packet. The connection only closes when an **ACK** has been sent by the other end and received by the initiator.
Maintaining a TCP connection requires the stations to remember a number of different parameters such as port numbers and sequence numbers. Each connection has this set of variables located in a **Transmission Control Block (TCB)**.
TCP knows whether the network TCP socket connection is opening, synchronizing, established by using the **SYN**chronize and ACKnowledge messages when establishing a network TCP socket connection.



When the communication between two computers ends, another 3-way communication is performed to tear down the TCP socket connection. This setup and teardown of a TCP socket connection is part of what qualifies TCP a reliable protocol. TCP also acknowledges that data is successfully received and guarantees the data is reassenbled in the correct order.

---

5. TCP operates over a 1-Gbps link. [MAY/JUNE 2013]

a) How long would it take for the TCP sequence numbers to wrap around completely?

b) Suppose an added 32-bit timestamp field increments 1000 times during the wraparound time you found out above. How long would it take for the timestamp to wrap around?

Answer:

a) First, we convert the speed of the link to the unit of bytes:
1 Gbps = (1 Gbits / 8 bits) = 125 MB/sec.
The sequence numbers wrap around when we have sent 2^{32} B worth of segments = 4 GB. On the link given in the question,
the wrap-around would take 4 GB / (125 MB/sec) = 32 seconds.
b) Incrementing every 32 ms (32 secs/1000), it would take about 32 x 4 x 10^9 ms, or about four years, for the timestamp field to wrap.

6. Explain TCP adaptive control and its uses in detail. [April – 17]

Nagle's Algorithm

Nagle introduced an elegant *self-clocking* solution.

The idea is that as long as TCP has any data in flight, the sender will eventually receive an ACK. This ACK can be treated like a timer firing, triggering the transmission of more data. Nagle‗s algorithm provides a simple, unified rule for deciding when to transmit:

When the application produces data to send if both the available data and the window ≥ MSS send a full segment else if there is un ACKed data in flight buffer the new data until an ACK arrives else send all the new data now

The socket interface allows the application to turn off Nagel‗s algorithm by setting the TCP NODELAY option. Setting this option means that data is transmitted as soon as possible.

Because TCP guarantees the reliable delivery of data, it retransmits each segment if an ACK is not received in a certain period of time. TCP sets this timeout as a function of the RTT it expects between the two ends of the connection.

Unfortunately, given the range of possible RTTs between any pair of hosts in the Internet, as well as the variation in RTT between the same two hosts over time, choosing an appropriate timeout value is not that easy. To address this problem, TCP uses an adaptive retransmission mechanism..

Original Algorithm

The idea is to keep a running average of the RTT and then to compute the timeout as a function of this RTT. Specifically, every time TCP sends a data segment, it records the time.

When an ACK for that segment arrives, TCP reads the time again, and then takes the difference between these two times as a Sample RTT.

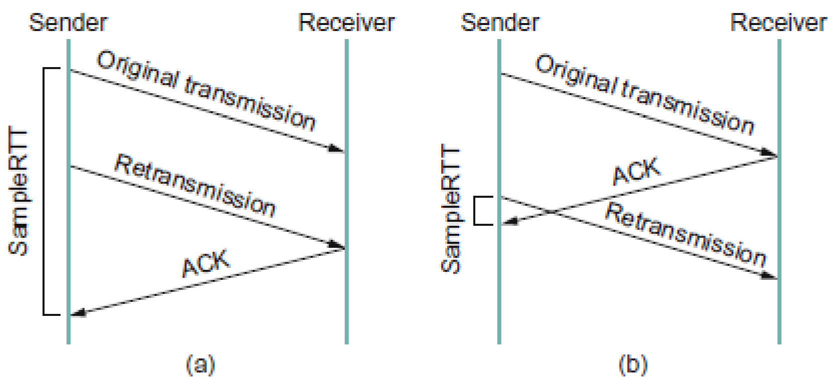
A small tracks changes in the RTT but is perhaps too heavily influenced by temporary fluctuations. The original TCP specification recommended a setting of between 0.8 and 0.9. TCP then uses Estimated RTT to compute the timeout in a rather conservative way:

Time Out = 2×EstimatedRTT

Karn/Partridge Algorithm

After several years of use on the Internet, a rather obvious flaw was discovered in this simple algorithm. The problem was that an ACK does not really acknowledge a transmission; it actually acknowledges the receipt of data.

In other words, whenever a segment is retransmitted and then an ACK arrives at the sender, it is impossible to determine if this ACK should be associated with the first or the second transmission of the segment for the purpose of measuring the sample RTT. It is necessary to know which transmission to associate it with so as to compute an accurate Sam- pleRTT. As illustrated in the figure below, if you assume that the ACK is for the original transmission but it was really for the second, then the Sample RTT is too large (a); if you assume that the ACK is for the second transmission but it was actually for the first, then the Sample RTT is too small (b).



Jacobson/Karels Algorithm

The main problem with the original computation is that it does not take the variance of the sample RTTs into account. Intuitively, if the variation among samples is small, then the Estimated RTT can be better trusted and there is no reason for multiplying this estimate by 2 to compute the timeout.

On the other hand, a large variance in the samples suggests that the timeout value should not be too tightly coupled to the Estimated RTT. In the new approach, the sender measures a new Sample RTT as before. It then folds this new sample into the timeout calculation as follows:

Difference = SampleRTT− EstimatedRTT EstimatedRTT = EstimatedRTT + (δ×Difference) Deviation = Deviation+ δ (Difference|−Deviation) where δ is a fraction between 0 and 1.

TCP then computes the timeout value as a function of both Estimated- RTT and Deviation as follows: TimeOut = $\mu \times$ EstimatedRTT + $\phi \times$ Deviation
The amount of data that is to be sent to the remote peer on a specific connection is controlled by two concurrent mechanisms:
The congestion in the network - The degree of network congestion is inferred by the calculation of changes in Round Trip Time (*RTT*): that is the amount of delay attributed to the network. This is measured by computing how long it takes a packet to go from sender to receiver and back to the client. This figure is actually calculated using a running smoothing algorithm due to the large variances in time.

The *RTT* value is an important value to determine the *congestion window*, which is used to control the amount of data sent out to the remote client. This provides information to the sender on how much traffic should be sent to this particular connection based on network congestion.
Client load - The rate at which the client can receive and process incoming traffic. The client sends a *receive window* that provides information to the sender on how much traffic should be sent to this connection based on client load.

---

6. Explain the various fields of the TCP header and working of the TCP protocol.

[Nov 16, April / May 2015,17]

TCP Header and working of TCP protocol;

Source port: 16 Bit number which identifies the Source Port number (Sending Computer's TCP Port).

Destination port: 16 Bit number which identifies the Destination Port number (Receiving Port).

Sequence number: 32 Bit number used for byte level numbering of TCP segments. If you are using TCP, each byte of data is assigned a sequence number. If SYN flag is set (during the initial three way handshake connection initiation), then this is the initial sequence number. The sequence number of the actual first data byte will then be this sequence number plus 1. For example, let the first byte of data by a device in a particular TCP header will have its sequence number in this field 50000. If this packet has 500 bytes of data in it, then the next packet sent by this device will have the sequence number of 50000 + 500 + 1 = 50501.

Acknowledgment Number: 32 Bit number field which indicates the next sequence number that the sending device is expecting from the other device.

Header Length: 4 Bit field which shows the number of 32 Bit words in the header. Also known as the Data Offset field. The minimum size header is 5 words (binary pattern is 0101).

Reserved: Always set to 0 (Size 6 bits).

Control Bit Flags: We have seen before that TCP is a Connection Oriented Protocol. The meaning of Connection Oriented Protocol is that, before any data can be transmitted, a reliable connection must be obtained and acknowledged. Control Bits govern the entire process of connection establishment, data transmissions and connection termination. The control bits are listed as follows: They are:

URG: Urgent Pointer.

ACK: Acknowledgement.

PSH: This flag means Push function. Using this flag, TCP allows a sending application to specify that the data must be pushed immediately. When an application requests the TCP to push data, the TCP should send the data that has accumulated without waiting to fill the segment.

RST: Reset the connection. The RST bit is used to RESET the TCP connection due to unrecoverable errors. When an RST is received in a TCP segment, the receiver must respond by immediately terminating the connection. A RESET causes both sides immediately to release the connection and all its resources. As a result, transfer of data ceases in both directions, which can result in loss of data that is in transit. A TCP RST indicates an abnormal terminination of the connection.

SYN: This flag means synchronize sequence numbers. Source is beginning a new counting sequence. In other words, the TCP segment contains the sequence number of the first sent byte (ISN).

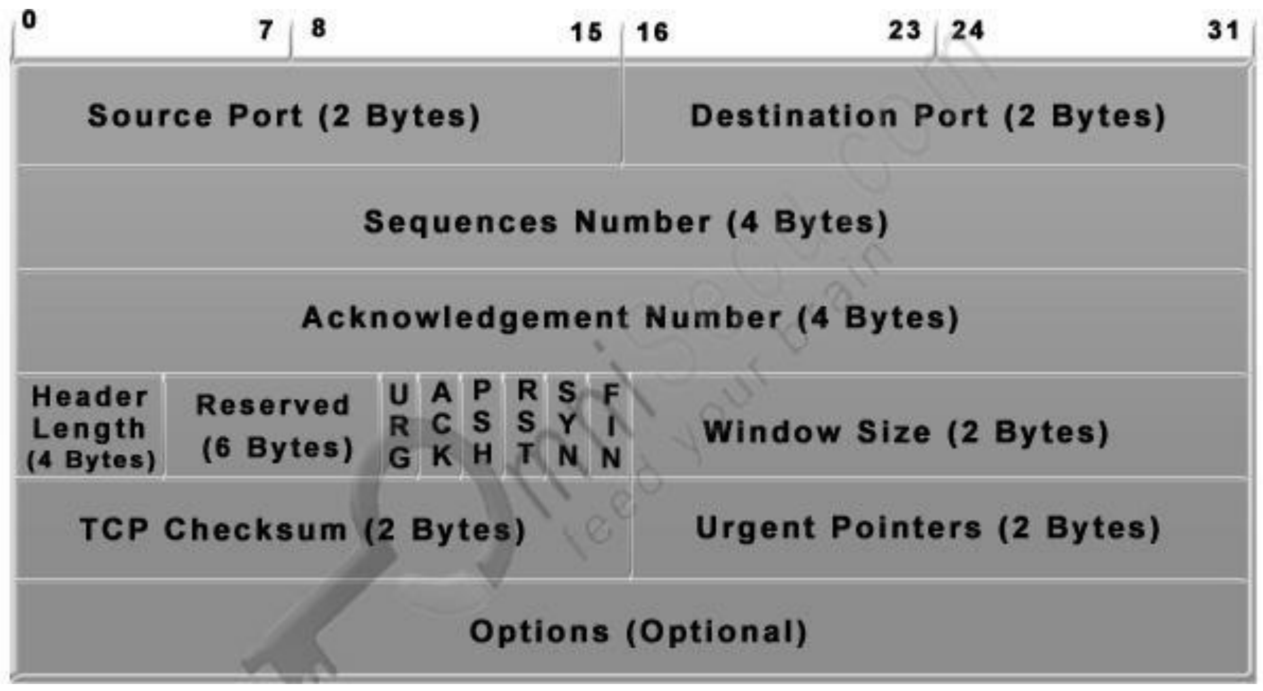
FIN: No more data from the sender. Receiving a TCP segment with the FIN flag does not mean that transferring data in the opposite direction is not possible.

Because TCP is a fully duplex connection, the FIN flag will cause the closing of connection only in one direction. To close a TCP connection gracefully, applications use the FIN flag.

Window: indicates the size of the receive window, which specifies the number of bytes beyond the sequence number in the acknowledgment field that the receiver is currently willing to receive.

Checksum: The 16-bit checksum field is used for error-checking of the header and data.

Urgent Pointer: Shows the end of the urgent data so that interrupted data streams can continue. When the URG bit is set, the data is given priority over other data streams (Size 16 bits).



7. Define UDP. Discuss its operation. Explain UDP Checksum with one example.

[May / June -16]

User Datagram Protocol (UDP): UDP is a connectionless transport layer protocol: each output operation by an application produces exactly one UDP datagram, which in turn causes one IP datagram to be sent. This is different from a stream oriented protocol such as TCP (see below), where the amount of data written by an application has little to do with what actually gets sent in a single IP datagram.

The UDP layer is responsible for communicating between two applications within two host computers; each application has a 16 bit port number assigned to it. There is reserved port number for various applications (see Section 12.9 of Comer). On the other hand, the IP layer only provides communication between the two host computers, and there can be multiple applications running on each computer.

UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, but there is no guarantee that they ever reach their destination. This is again unlike the TCP.

Format of UDP messages:

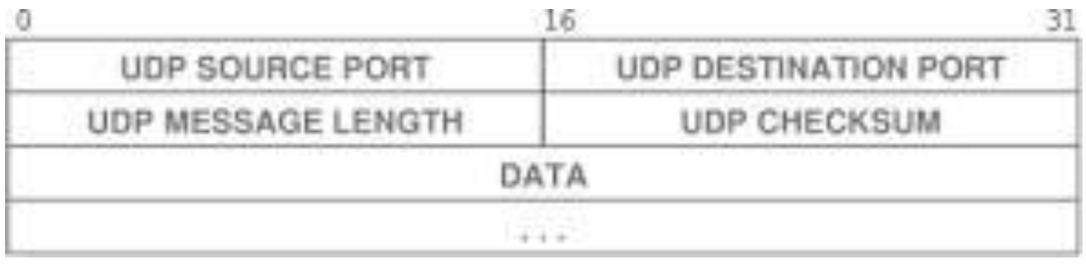
Each UDP message is called a UDP datagram (just like an IP datagram, since UDP is after all an extension of IP to provide communication between two applications). Each UDP datagram has two parts: a UDP header and a UDP data area. The header is divided into the following four 16 bit fields

1. Source port number (16 bits): This contains the 16 bit UDP protocol source port number. This port number is optional; if used it specifies the port to which replies should be sent; if not used, it should be zero.

2. Destination port number (16 bits): This contains the destination port number, and it is used to demultiplex datagrams among the various processes waiting to receive them in the destination computer.

3. UDP message length (16 bits): This is a count of bytes in the UDP datagram, and includes the length of the UDP header and data (unlike IP which includes only the header length). The minimum value is 8, the length of the header alone.

4. UDP checksum (16 bits): This is optional; a value of zero indicates that the checksum has not been computed. The UDP checksum covers more information than is present in the UDP datagram alone. To compute the checksum, UDP prepends a pseudo-header to the UDP datagram, appends an octet of zeros to pad the datagram to an exact multiple of 16 bits, and computes the checksum over the entire object. The pseudo header is shown in Figure 12.2 of Comer, and includes the 32 bit source and destination IP addresses, a 8 bit protocol field which is 17 for UDP, and a 16 bit UDP length (which is also present in the UDP header). To compute a checksum, the UDP software computes the 16 bit one‘s complement sum of the pseudo header, UDP header, and user data, and takes its 1s complement. If the calculated checksum is 0, it is stored as all one bits (65535), which is equivalent in one‘s complement arithmetic. This is to distinguish it from 0 which implies that the checksum was not computed. The purpose of prepending a pseudo header to the UDP header before computing the checksum is to ensure that the UDP datagram has reached its correct destination, since this requires the correct destination IP address in addition to the correct destination port number.



UDP Checksum:

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- In other words, all 16-bit words are summed using one's complement arithmetic. The sum is then one's complemented to yield the value of the UDP checksum field.
- If the checksum calculation results in the value zero (all 16 bits 0) it should be sent as the one's complement(all 1s).
- The difference between IPv4 and IPv6 is in the data used to compute the checksum.

PART –B

1. Explain the message transfer using Simple Mail Transfer Protocol. & explain the final delivery of email to the end user using POP3 [APR/MAY-15,16, 17] Describe the message format and the message transfer and the underlying protocol involved in the working of the electronic mail. [NOV/DEC-2013], [NOV/DEC-2012, 15,16]

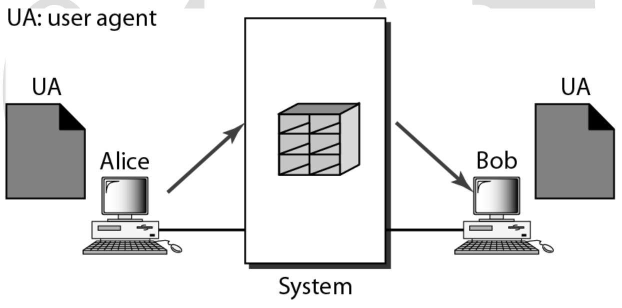
One of the most popular Internet services is electronic mail (e-mail). We first study the general architecture of an e-mail system including the three main components: user agent, message transfer agent, and message access agent. We then describe the protocols that implement these components.

Architecture.

To explain the architecture of e-mail, we give four scenarios

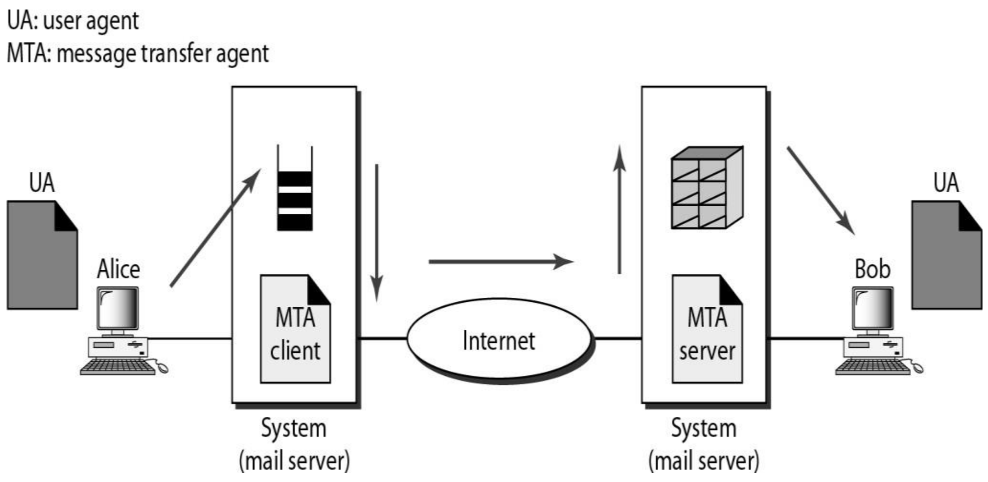
First Scenario

In the first scenario, the sender and the receiver of the e-mail are 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 .When the sender and the receiver of an e-mail are on the same system, we need only two user agents.



Second Scenario

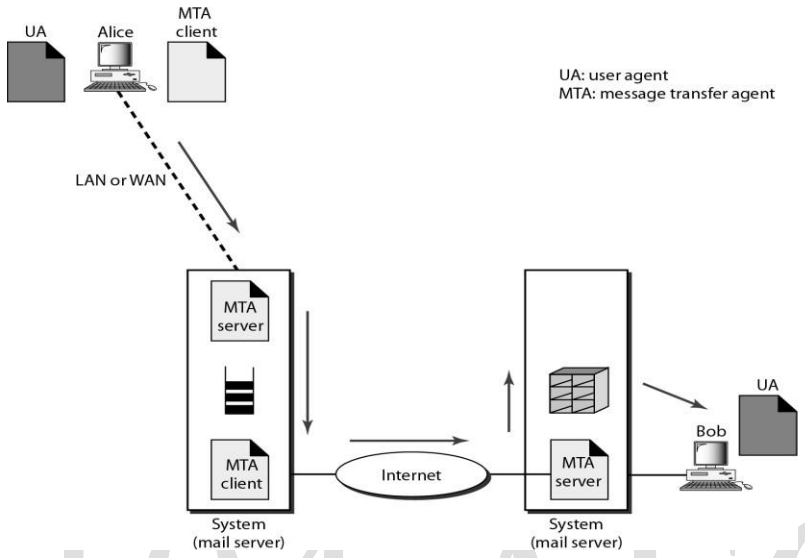
In the second scenario, the sender and the receiver of the e-mail are users on two different systems. The message needs to be sent over the Internet



When the sender and the receiver of an e-mail are on different systems, we need two UAs and a pair of MTAs (client and server).

Third Scenario

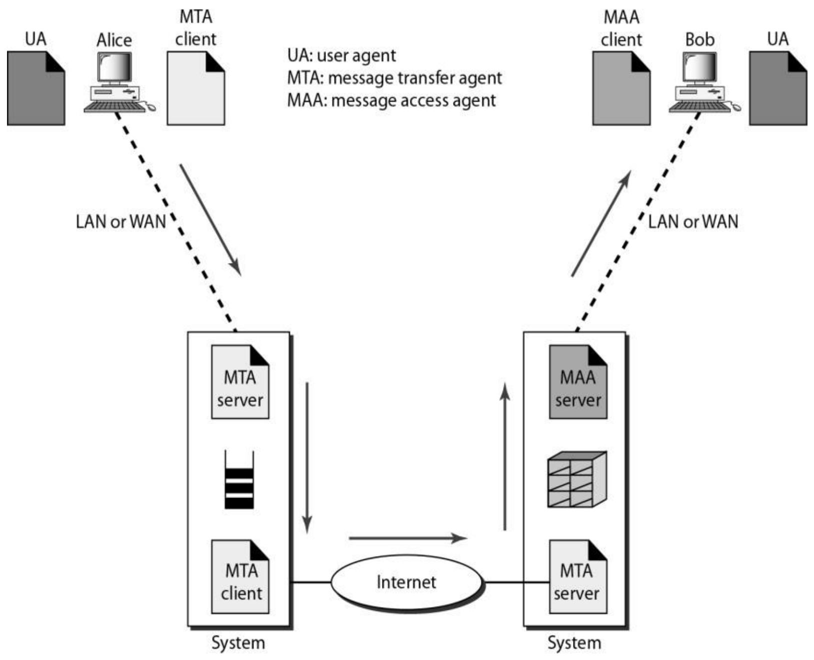
In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for handling e-mails-all users need to send their messages to this mail server



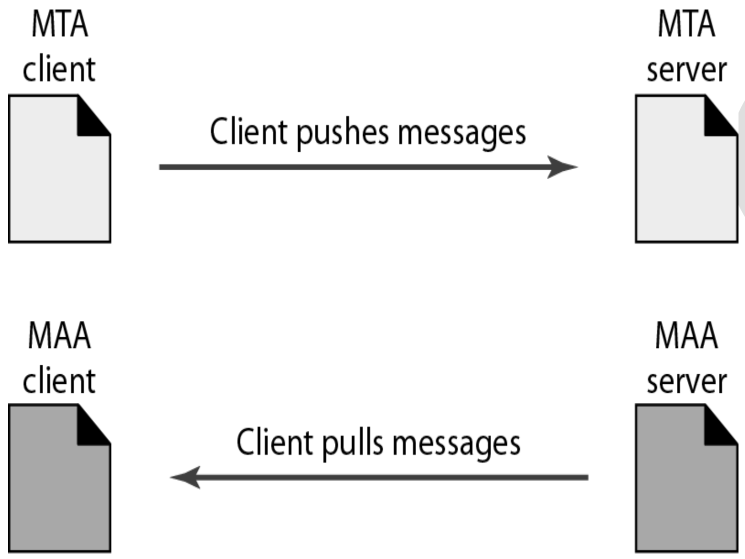
When the sender is connected to the mail server via a LAN or a WAN, we need two *UAs* and two pairs of MTAs (client and server).

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.



When both sender and receiver are connected to the mail server via a LAN or a WAN,we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server).



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.

User Agent Types

There are two types of user agents: command-driven and GUI-based.

a) Command-Driven

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.*

b) GUI-Based

Modern 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.

Sending Mail

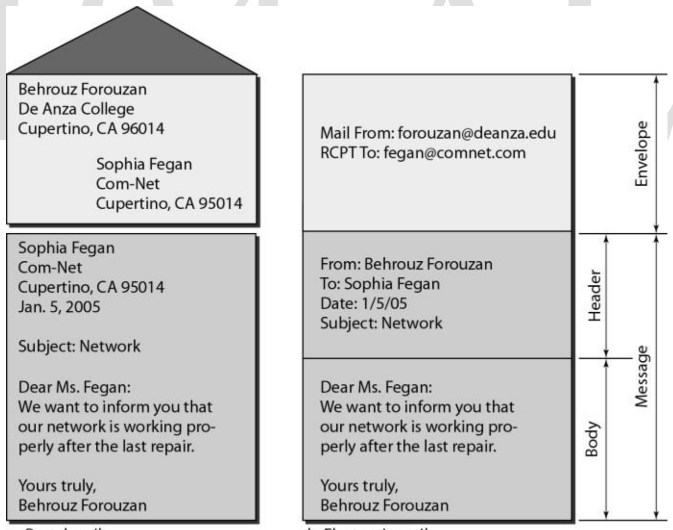
To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message

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.



Behrouz Forouzan
De Anza College
Cupertino, CA 96014

Sophia Fegan
Com-Net
Cupertino, CA 95014

Sophia Fegan
Com-Net
Cupertino, CA 95014
Jan. 5, 2005

Subject: Network

Dear Ms. Fegan:
We want to inform you that our network is working properly after the last repair.

Yours truly,
Behrouz Forouzan

a. Postal mail

Mail From: forouzan@deanza.edu
RCPT To: fegan@comnet.com

From: Behrouz Forouzan
To: Sophia Fegan
Date: 1/5/05
Subject: Network

Dear Ms. Fegan:
We want to inform you that our network is working properly after the last repair.

Yours truly,
Behrouz Forouzan

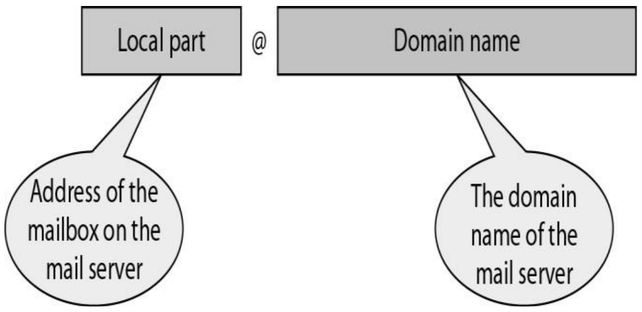
b. Electronic mail

Receiving Mail.

The user agent is triggered by the user (or a timer). If a user has mail, the *UA* informs the user with a notice.

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

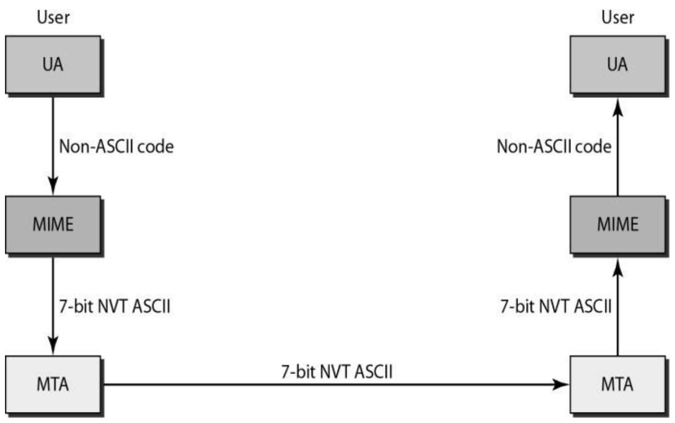

Local part
@
Domain name
Address of the mailbox on the mail server
The domain name of the mail server

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*.

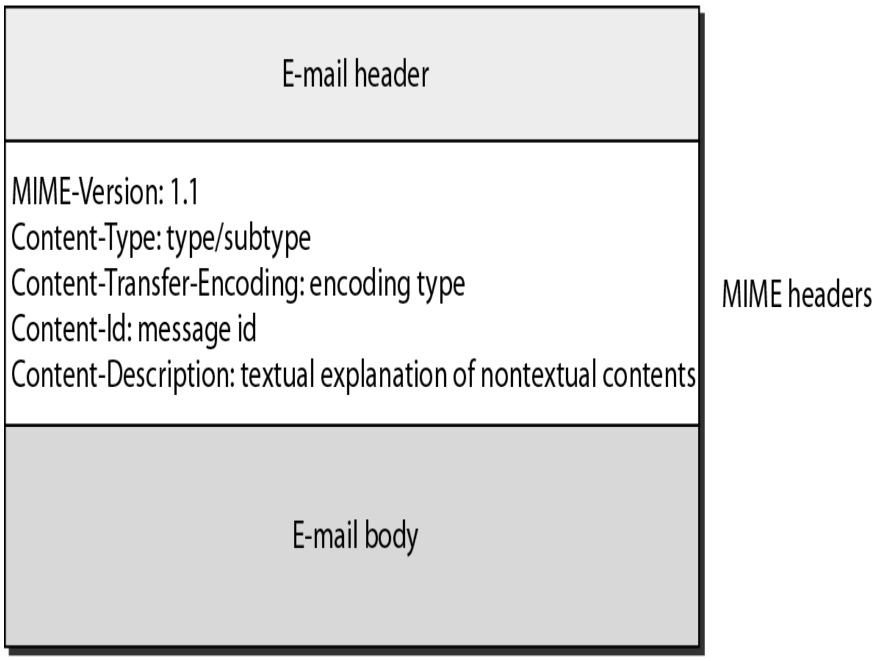
MIME

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.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.



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


E-mail header
MIME-Version: 1.1
Content-Type: type/subtype
Content-Transfer-Encoding: encoding type
Content-Id: message id
Content-Description: textual explanation of nontextual contents
MIME headers
E-mail body

1. MIME-Version This header defines the version of MIME used. The current version is 1.1.

2. Content-Type This header defines the type of data used in the body of the message.

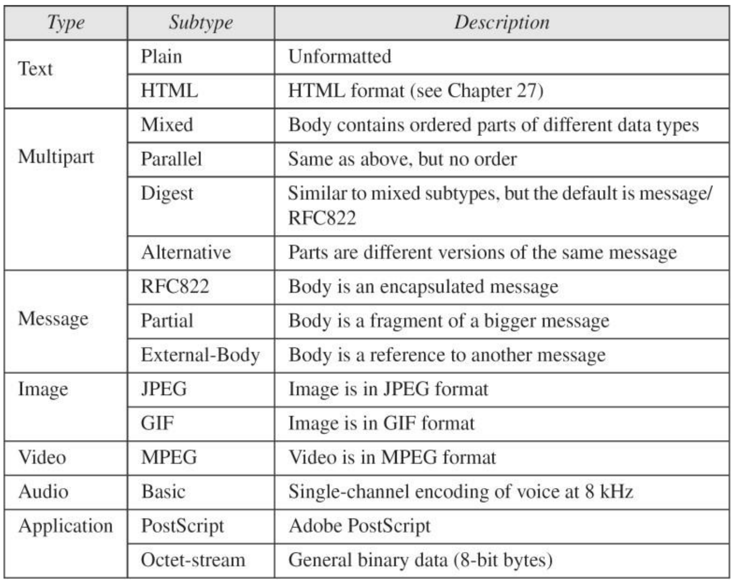
The content type and the content subtype are separated by a slash

MIME allows seven different types of data.

3. Content-Transfer-Encoding .This header defines the method used to encode the messages into O s and 1s for transport

4. Content-Id This header uniquely identifies the whole message in a multiple-message environment

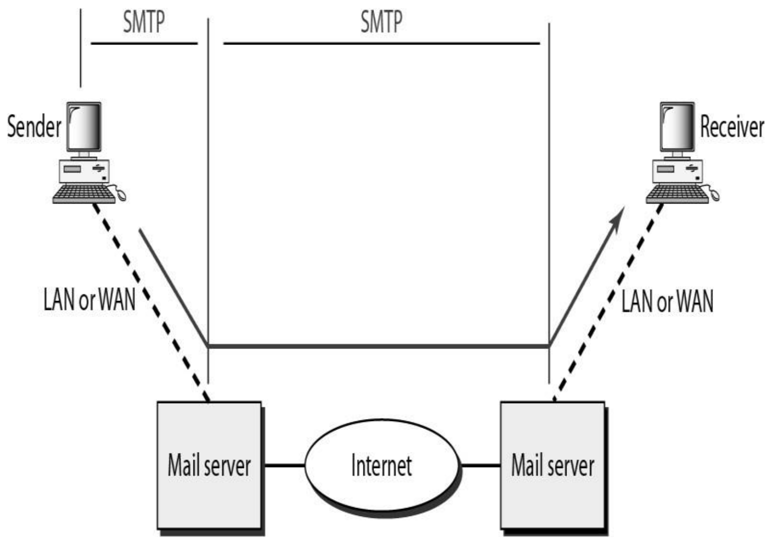
5. Content-Description This header defines whether the body is image, audio, or video.



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	JPEG	Image is in JPEG 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)

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).



Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server

Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Commands

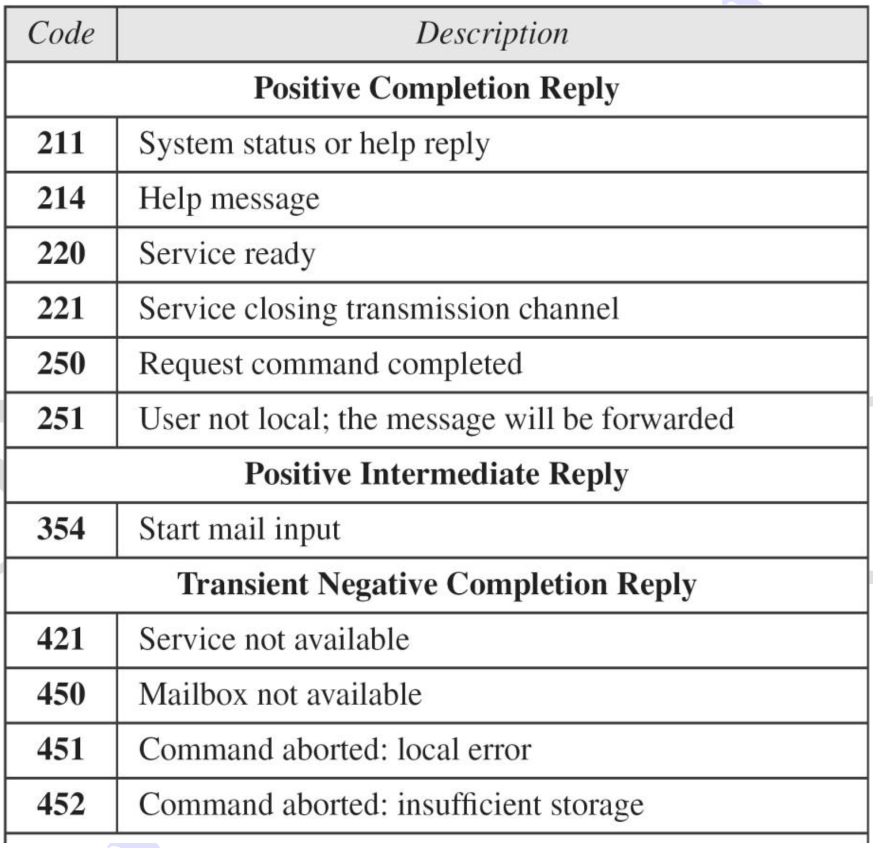
Commands are sent from the client to the server. It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands.

KEYWORD : argument(s)

Keyword	*Argument(s)*
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPT TO	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
SEND FROM	Intended recipient of the message
SMOL FROM	Intended recipient of the message
SMAL FROM	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.



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

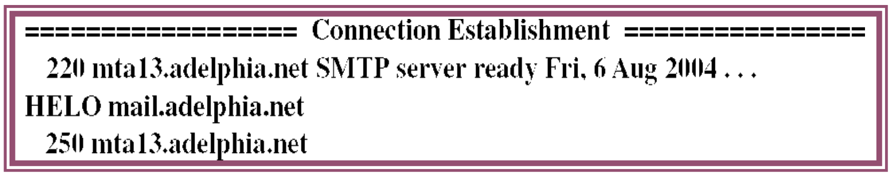
Mail Transfer Phases

The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.

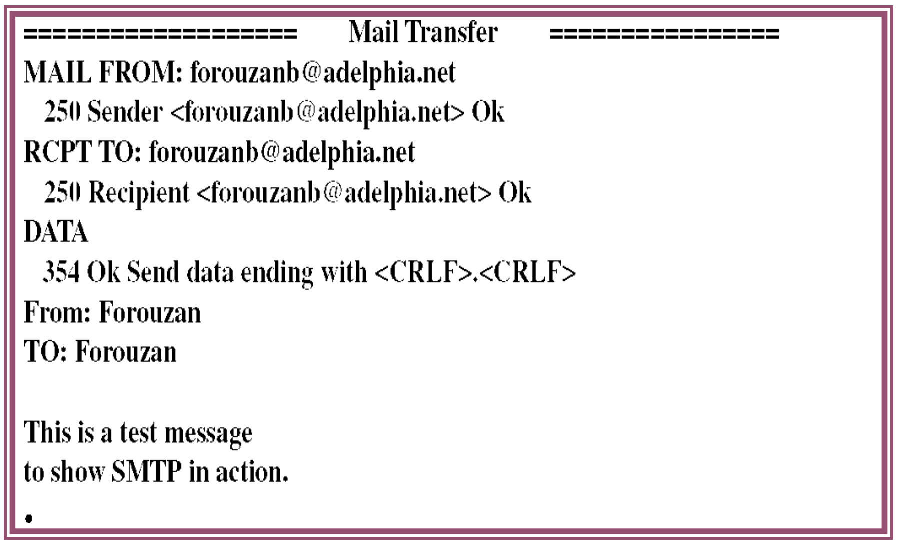
$ telnet mail.adelphia.net 25

Trying 68.168.78.100 . . .

Connected to mail.adelphia.net (68.168.78.100).

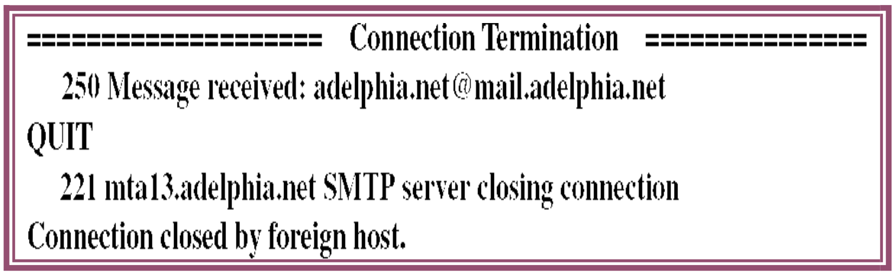


```
=================  Connection Establishment  ================
   220 mta13.adelphia.net SMTP server ready Fri, 6 Aug 2004 . . .
HELO mail.adelphia.net
   250 mta13.adelphia.net
```



```
===================      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).

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

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.

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. 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.

---

2. Write short notes on Web services. [MAY/JUNE-2016] [APR/MAY-2015]

An ordering application at enterprise A would send a message to an order fulfilment application at enterprise B, which would respond immediately indicating whether the order can be filled.

Perhaps, if the order cannot be filled by B, the application at A would immediately order from another supplier, or solicit bids from a collection of suppliers.

In the business world, enabling applications to interact directly with each other is called business-to-business (B2B) integration when the applications are at different enterprises and enterprise application integration.

Custom Application Protocols (WSDL, SOAP)

The architecture informally referred to as SOAP is based on *Web Services Description Language (WSDL)* and *SOAP*. Both of these standards are issued by the World Wide Web Consortium (W3C). This is the architecture that people usually mean when they use the term Web Services. Both WSDL and SOAP consist primarily of a protocol specification language.

Both languages are based on XML with an eye toward making specifications accessible to software tools such as stub compilers and directory services.

Defining Application Protocols

WSDL has chosen a procedural *operation* model of application protocols. An abstract web service interface consists of a set of named operations, each representing a simple interaction between a client and the web service.

An operation is analogous to a remotely callable procedure in an RPC system. An example from W3C‘s WSDL Primer is a hotel reservation web service with two operations, Check Availability and Make Reservation.

Each operation specifies a *message exchange pattern (MEP)* that gives the sequence in which the messages are to be transmitted, including the fault messages to

be sent when an error disrupts the message flow.
Several MEPs are predefined and new custom MEPs can be defined, but it appears that in practice only two MEPs are being used:

In-Only (a single message from client to service) and In-Out (a request from client and corresponding reply from service). These patterns should be very familiar, and suggest that the costs of supporting MEP flexibility perhaps outweigh the benefits.

MEPs are templates that have placeholders instead of specific message types or formats, so part of the definition of an operation involves specifying which message formats to map into the placeholders in the pattern.

Message formats are defined as an abstract data model using XML Schema. XML Schema provides a set of primitive data types and ways to define compound data types.
Data that conforms to an XML Schema-defined format—its abstract data model—can be concretely represented using XML, or it can use another representation, such as the —binary‖ representation Fast

Defining Transport Protocols

SOAP uses many of the same strategies as WSDL, including message formats defined using XML Schema, bindings to underlying protocols, MEPs, and reusable specification elements identified using XML namespaces.

SOAP is used to define transport protocols with exactly the features needed to support a particular application protocol. SOAP aims to make it feasible to define many such protocols by using reusable components.

Each component captures the header information and logic that go into implementing a particular feature. To define a protocol with a certain set of features, just compose the corresponding components.
A SOAP feature is an extension of the SOAP messaging framework. A SOAP feature specification must include:

- A URI that identifies the feature;
- The state information and processing, abstractly described, that is required at each SOAP node to implement the feature;
- The information to be relayed to the next node;
- If the feature is a MEP, the life cycle and temporal/causal relationships of the messages

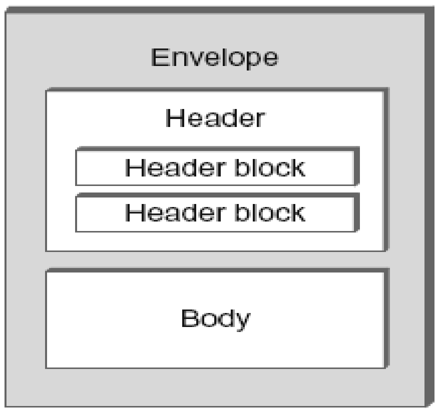


Fig: SOAP message structure

exchanged (e.g., responses follow requests and are sent to the originator of the request).

There are two strategies for defining a SOAP protocol that will implement them.

For example, we could obtain a request-response protocol by binding SOAP to HTTP, with a SOAP request in an HTTP request, and a SOAP reply in an HTTP response. new bindings may be defined using the SOAP Protocol Binding Framework.

The more interesting way to implement features involves *header blocks*. A SOAP message consists of an envelope, which contains a header that contains header blocks, and a body that contains the payload destined for the ultimate receiver.

A Generic Application Protocol (REST)

The WSDL/SOAP Web Services architecture is based on the assumption that the best way to integrate applications across networks is via protocols that are customized to each application.

The REST Web Services architecture is based on the assumption that the best way to integrate applications across networks is by applying the model underlying the World Wide Web architecture. This model, articulated by Web architect Roy Fielding, is known as *RE presentational State Transfer (REST)*.

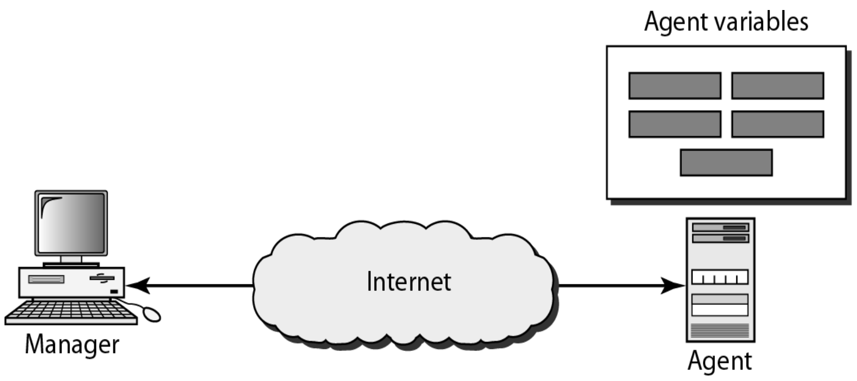
An area where WSDL/SOAP may have an advantage is in adapting or wrapping previously written, legacy applications to conform to web services. This is an important point since most web services will be based on legacy applications for the near future at least.

These applications usually have a procedural interface that maps more easily into WSDL‘s operations than REST states.

The REST versus WSDL/SOAP competition may very well hinge on how easy or difficult it turns out to be to devise REST-style interfaces for individual web services. We may find that some web services are better served by WSDL/SOAP and others by REST.

3. Explain the SNMP protocol in detail. [MAY/JUNE-2014,15,17, Nov16]

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. SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers. SNMP is an application-level protocol in which a few manager stations control a set of agents. SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology



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. The manager can also make the router perform certain actions. The manager can 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.

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.

To do management tasks, SNMP uses two other protocols: Structure of Management Information (SMI) and Management Information Base (MIB).

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. The packets exchanged contain the object (variable) names and their status (values). SNMP is responsible for reading and changing these values

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 . We also need rules to define the type of the objects. SMI is a protocol that defines these rules. SMI is a collection of general rules to name objects and to list their types.

Role of MIB

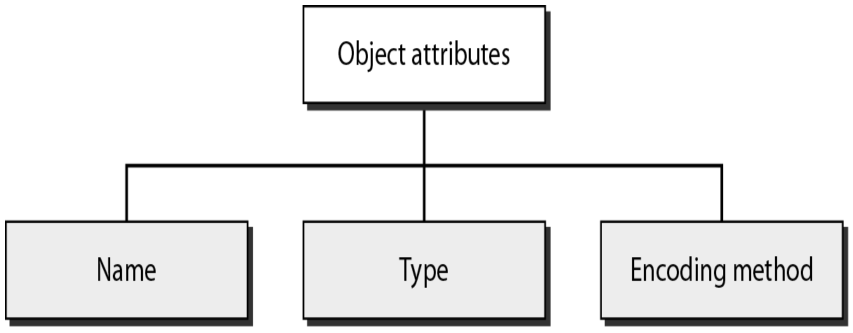
For each entity to be managed, this protocol must define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object. This protocol is MIB. MIB creates a set of objects defined for each entity similar to a database

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

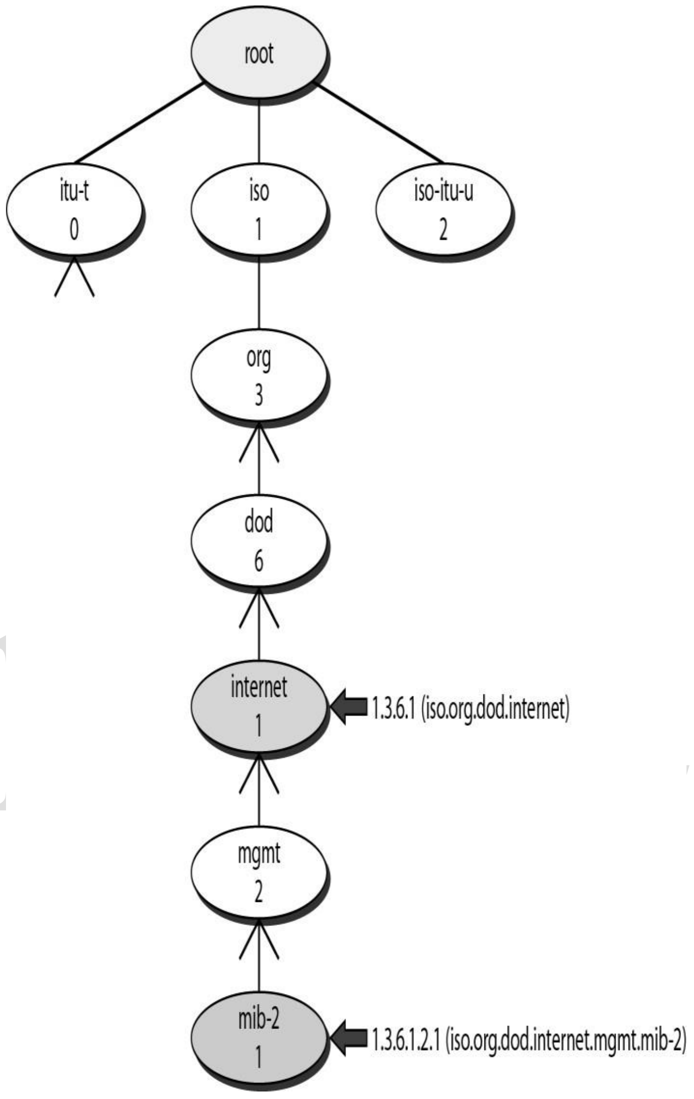
SMI is a guideline for SNMP. It emphasizes three attributes to handle an object: name, data type, and encoding method



Name

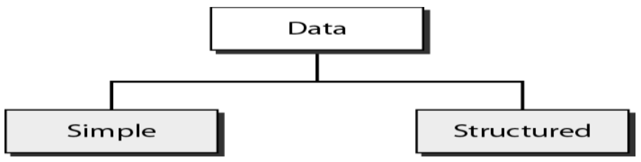
SMI requires that each managed object (such as a router, a variable in a router, a value) have a unique name. To name objects globally, SMI uses an object identifier,

which is a hierarchical identifier based on a tree structure. 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


root
itu-t
0
iso
1
iso-itu-u
2
org
3
dod
6
internet
1
1.3.6.1 (iso.org.dod.internet)
mgmt
2
mib-2
1
1.3.6.1.2.1 (iso.org.dod.internet.mgmt.mib-2)

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 SMI has two broad categories of data type: *simple* and *structured*



Simple Type

The simple data types are atomic data types. Some of them are taken directly from ASN. l; others are added by SMI.

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

SMI defines two structured data types: *sequence* and *sequence of*

Sequence.

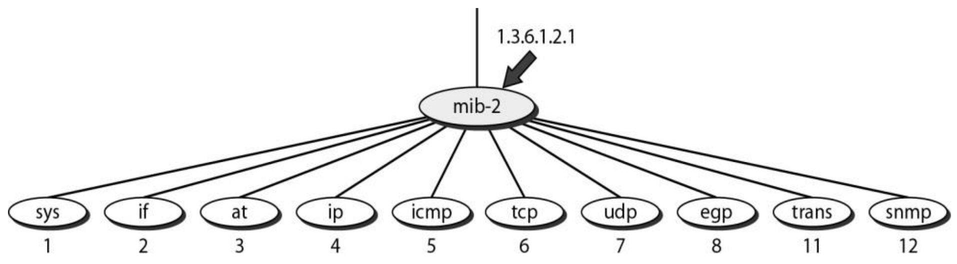
A *sequence* data type is a combination of simple data types, not necessarily of the same type. It is analogous to the concept of a *struct* or a *record* used in programming languages such as C.

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.

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

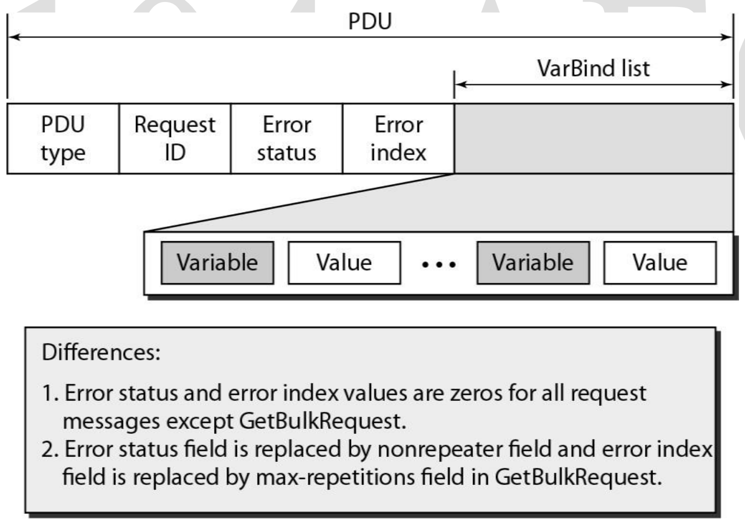


The following is a brief description of some of the objects:

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.

PDUs

SNMPv3 defines eight types of packets (or PDUs): GetRequest, GetNextRequest, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report



Differences:

1. Error status and error index values are zeros for all request messages except GetBulkRequest.
2. Error status field is replaced by nonrepeater field and error index field is replaced by max-repetitions field in GetBulkRequest.

PDU type. This field defines the type of the POD
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.
Non repeaters. 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.

---

4. Write short notes on DNS. [NOV/DEC-2015,16,MAY/JUNE-2014,17]
Discuss the need for name resolution. Illustrate the domain name hierarchy and the steps in resolution. [MAY/JUNE-2013],
[NOV/DEC-2014]
Explain in detail about name services [MAY/JUNE-2011]

The Domain Name System (DNS) is a supporting program that is used by other programs such as e-mail.

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.

Flat Name Space

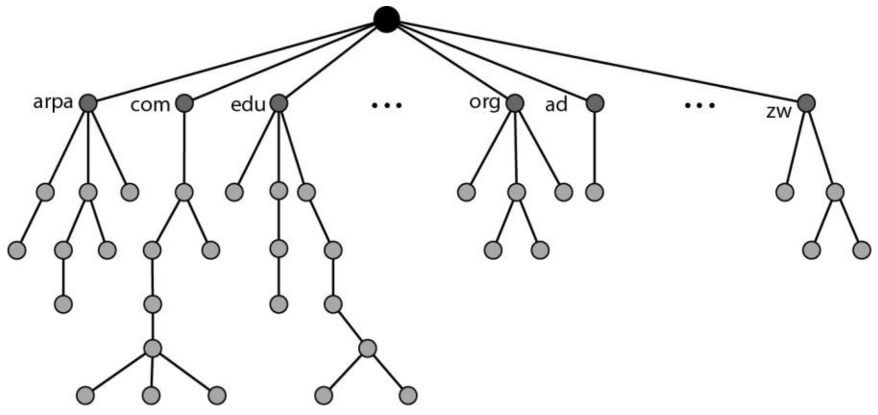
In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section.

Hierarchical Name 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.

DOMAIN NAME SPACE

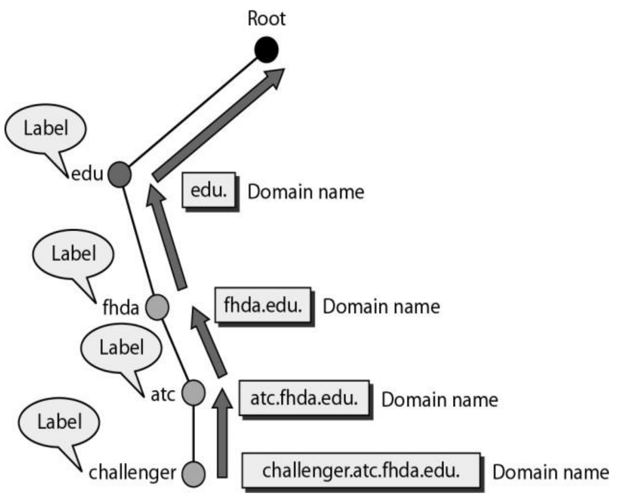
To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.



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

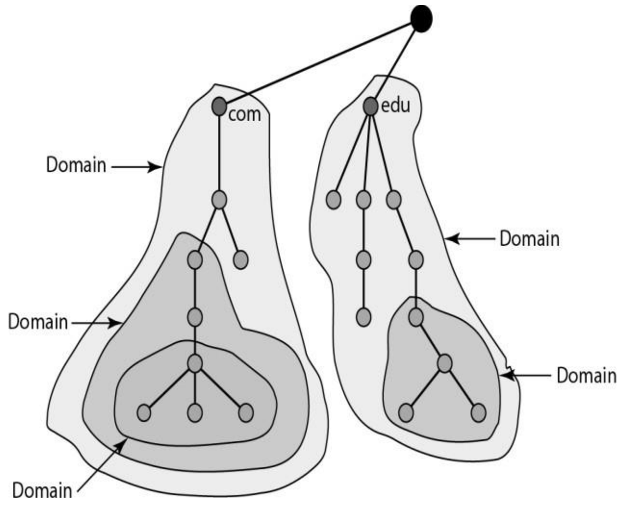
Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label,



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,
challenger.ate.tbda.edu.

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 challenger .

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.. Note that a domain may itselfbe divided into domains

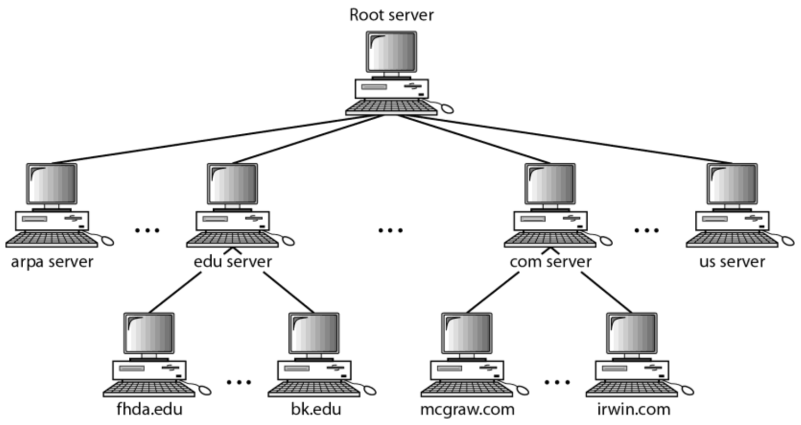


DISTRIBUTION OF NAME SPACE

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

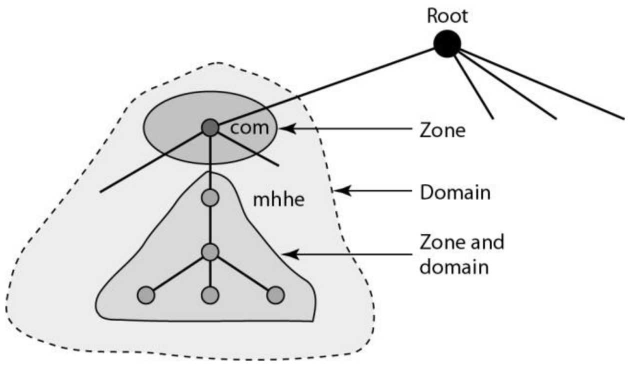
Hierarchy of Name Servers

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



Zone

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



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

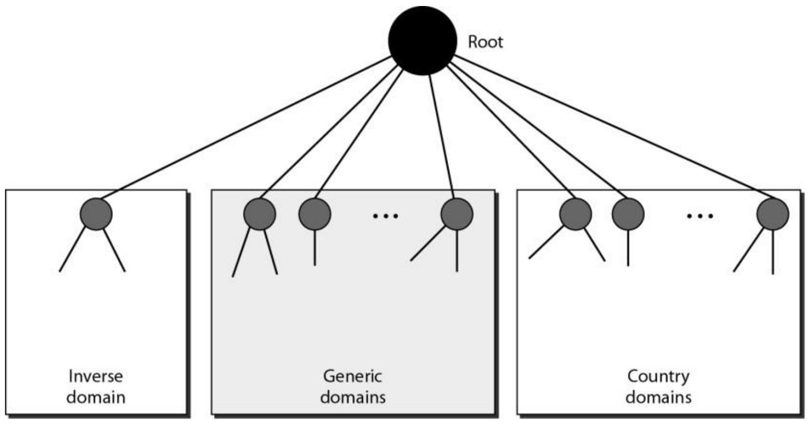
Primary and Secondary Servers

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.

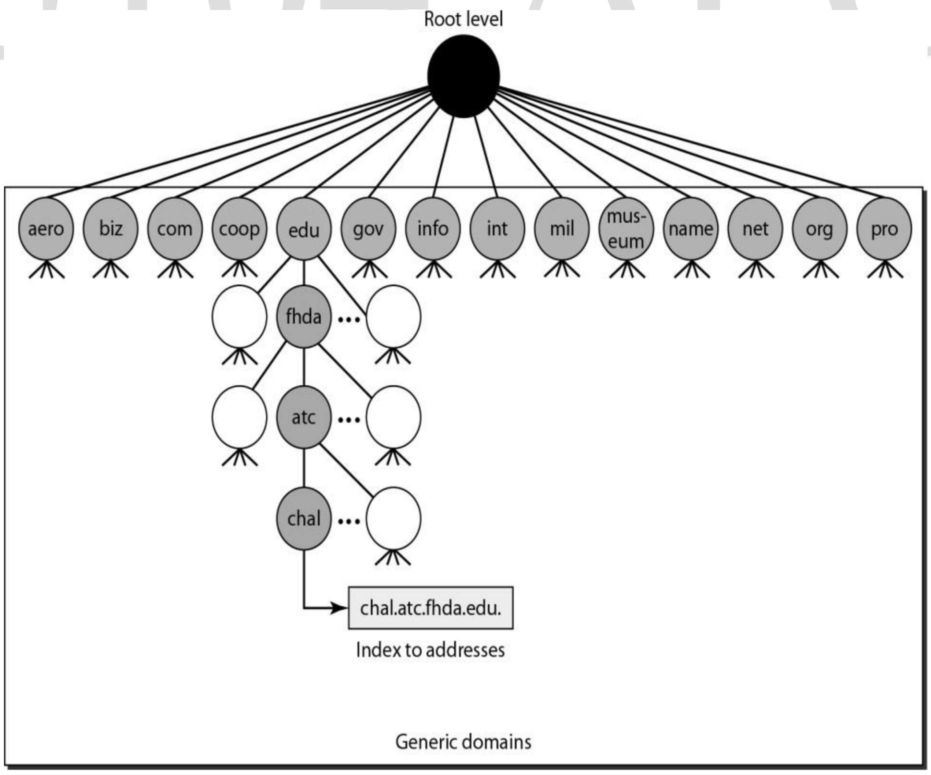
DNS IN THE INTERNET

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



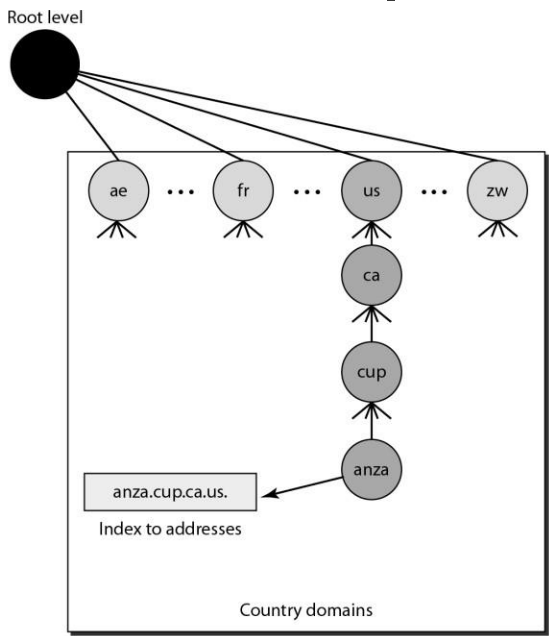
Generic Domains

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.



Country Domains The country domains section uses two-character country abbreviations (e.g., us for United States).

Inverse Domain The inverse domain is used to map an address to a name



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 information.

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.

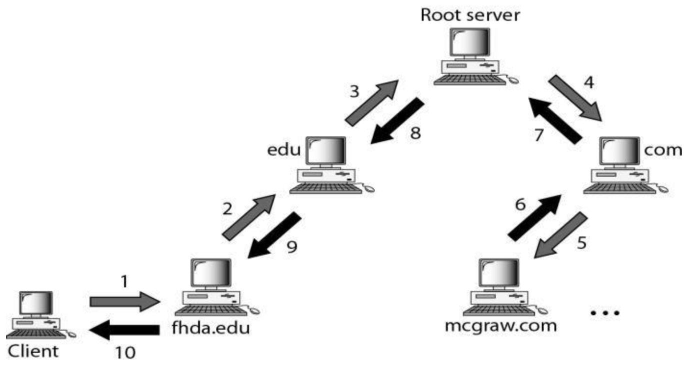
Mapping Addresses to Names

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

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.



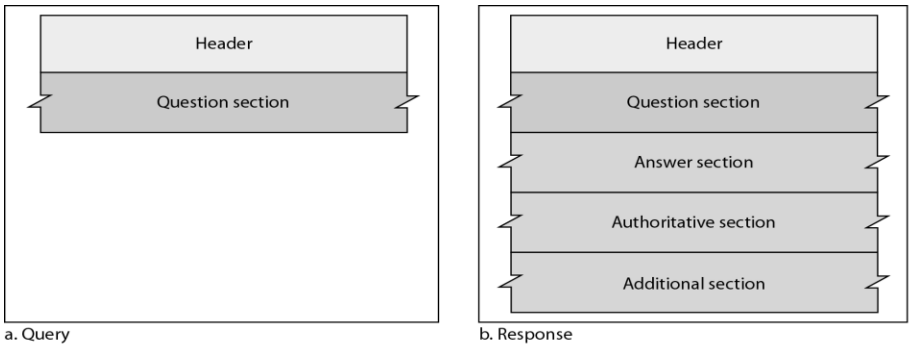
Caching

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..

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.



The header is 12 bytes

Question Section

This is a section consisting of one or more question records. It is present on both query and response messages.

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).

Authoritative Section

This is a section consisting of one or more resource records. It is present only on response messages.

Additional Information Section

This is a section consisting of one or more resource records. It is present only on response messages.

5.A Explain with example, the HTTP (World Wide Web). [NOV/DEC-2013], [NOV/DEC-2012],[MAY/JUNE-2016,17]

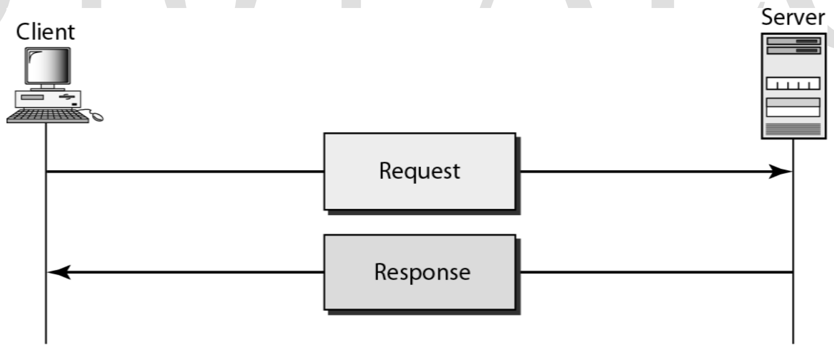
Explain the various process involved after typing the URL in the taskbar.

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 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 HTTP uses the services ofTCP on well-known port 80.

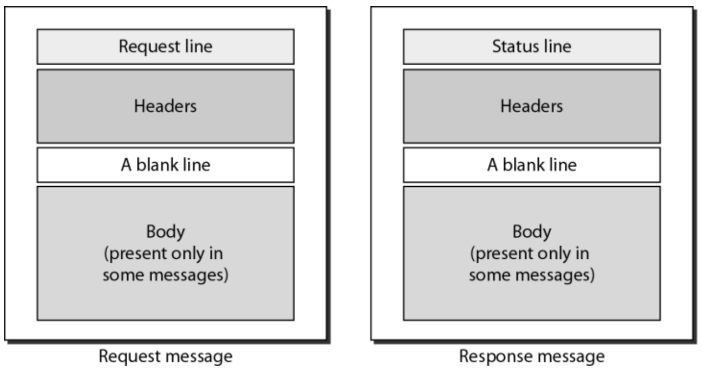
HTTP Transaction

Figure illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol

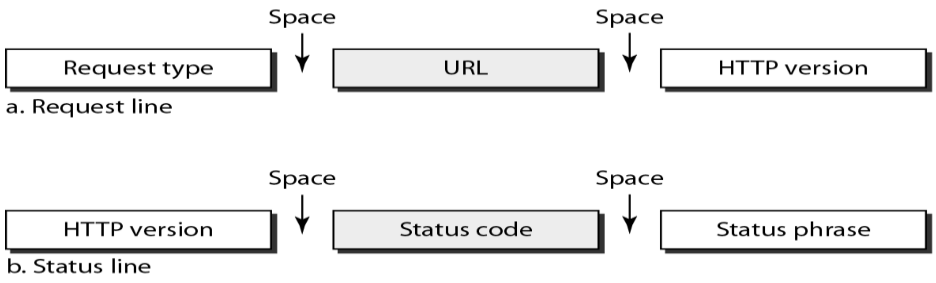


Messages

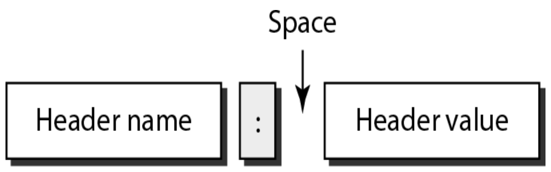
The formats of the request and response messages are similar; both are shown in Figure. A request message consists of a request line, a header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body



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



Request type. This field is used in the request message URL.
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
Status phrase. This field is used in the response message. It explains the status code in text form.
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



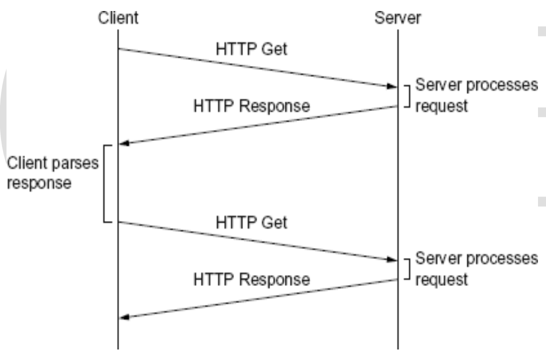
General header The general header gives general information about the message and can be present in both a request and a response
Request header The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format
Response header The response header can be present only in a response message. It specifies the server's configuration and special information about the request

Entity header The entity header gives infonnation about the body of the document.
Body The body can be present in a request or response message. Usually, it contains the document to be sent or received

Header	*Description*
Cache-control	Specifies information 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

Persistent Versus Non persistent Connection 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



Non persistent Connection

In a non persistent 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.

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.

5B. Explain in detail the Network Software and Network Performance (Nov – 16)

Network Software

Implementing Network Software: Network architectures and protocol specifications are essential things, but a good blueprint is not enough to explain the phenomenal success of the Internet: The number of computers connected to the Internet has grown exponentially for almost 3 decades. The number of users of the Internet was estimated to be around 1.8 billion in 2009—an impressive percentage of the world‘s population.

Application Programming Interface (Sockets): When implementing a network application which is the interface exported by the network. Since most network protocols are implemented in software, and nearly all computer systems implement their network protocols as part of the operating system, when we refer to the interface –exported by the network,‖ referring to the interface that the OS provides to its networking subsystem. This interface is often called the network application programming interface (API). The first step is to create a socket, which is done with the following operation:

int socket(int domain, int type, int protocol)

The reason that this operation takes three arguments is that the socket interface was designed to be general enough to support any underlying protocol suite. Specifically, the domain argument specifies the protocol family that is going to be used: PF INET denotes the Internet family, PF UNIX denotes the Unix pipe facility, and PF PACKET denotes direct access to the network interface (i.e., it bypasses the TCP/IP protocol stack).

The type argument indicates the semantics of the communication. SOCK STREAM is used to denote a byte stream. SOCK DGRAM is an alternative that denotes a message-oriented service, such as that provided by UDP. The protocol argument identifies the specific protocol that is going to be used. In our case, this argument is UNSPEC because the combination of PF INET and SOCK STREAM implies TCP.

The next step depends on whether you are a client or a server. On a server machine, the application process performs a *passive* open—the server says that it is prepared to accept connections, but it does not actually establish a connection. The server does this by invoking the following three operations:

int bind(int socket, struct sockaddr *address, int addr len)

int listen(int socket, int backlog)

int accept(int socket, struct sockaddr *address, int *addr len)

The bind operation, as its name suggests, binds the newly created socket to the specified address. This is the network address of the *local* participant the server.

The listen operation then defines how many connections can be pending on the specified socket. Finally, the accept operation carries out the passive open. It is a blocking operation that does not return until a remote participant has established a connection, and when it does complete it returns a *new* socket that corresponds to this just-established connection, and the address argument contains the *remote* participant‘s address.

On the client machine, the application process performs an active open; that is, it says who it wants to communicate with by invoking the following single operation:

int connect(int socket, struct sockaddr *address, int addr len)

This operation does not return until TCP has successfully established a connection, at which time the application is free to begin sending data. In this case, address contains the remote participant‘s address. In practice, the client usually specifies only the remote participant‘s address and lets the system fill in the local information. Whereas a server usually listens for messages on a well-known port, a client typically does not care which port it uses for itself; the OS simply selects an unused one. Once a connection is established, the application processes invoke the following two operations to send and receive data:

int send(int socket, char *message, int msg len, int flags)

int recv(int socket, char *buffer, int buf len, int flags)

The first operation sends the given message over the specified socket, while the second operation receives a message from the specified socket into the given buffer. Both operations take a set of flags that control certain details of the operation.

NETWORK PERFORMANCE

Like any computer system, however, computer networks are also expected to perform well. This is because the effectiveness of computations distributed over the network often depends directly on the efficiency with which the network delivers the computation‘s data. It is therefore important to understand the various factors that impact network performance.

Bandwidth and Latency

Network performance is measured in two fundamental ways: *bandwidth* (also called *throughput*) and *latency* (also called *delay*). The bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. For example, a network might have a bandwidth of 10 million bits/second (Mbps), meaning that it is able to deliver 10 million bits every second. It is sometimes useful to think of bandwidth in terms of how long it takes to transmit each bit of data.

The more sophisticated the transmitting and receiving technology, the narrower each bit can become and, thus, the higher the bandwidth. For logical process-to-process channels, bandwidth is also influenced by other factors, including how many times the software that implements the channel has to handle, and possibly transform, each bit of data.

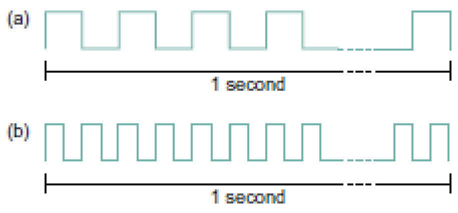


Fig: Bits transmitted at a particular bandwidth can be regarded as having some width: (a) bits transmitted at 1 Mbps (each bit is 1µs wide); (b) bits transmitted at 2 Mbps (each bit is 0.5µs wide).

The second performance metric, latency, corresponds to how long it takes a message to travel from one end of a network to the other. Latency is measured strictly in terms of time. For example, a transcontinental network might have a latency of 24 milliseconds (ms); that is, it takes a message 24 ms to travel from one coast of North America to the other. There are many situations in which it is more important to know how long it takes to send a message from one end of a network to the other and back, rather than the one-way latency. This is called the round-trip time (RTT) of the network.

Latency is having three components. First, there is the speed-of-light propagation delay. This delay occurs because nothing, including a bit on a wire, can travel faster than the speed of light. Second, there is the amount of time it takes to transmit a unit of data. This is a function of the network bandwidth and the size of the packet in which the data is carried. Third, there may be queuing delays inside the network, since packet switches generally need to store packets for some time before forwarding them on an outbound link. So, the total latency is defined as

Latency = Propagation+Transmit+Queue

Propagation = Distance/SpeedOfLight

Transmit = Size/Bandwidth

Where Distance is the length of the wire over which the data will travel, Speed Of Light is the effective speed of light over that wire, Size is the size of the packet, and Bandwidth is the bandwidth at which the packet is transmitted. Bandwidth and latency combine to define the performance characteristics of a given link or channel.

Delay×Bandwidth Product

It is also useful to talk about the product of these two metrics, often called the *delay ×bandwidth product*. Intuitively, if we think of a channel between a pair of processes as a hollow pipe, where the latency corresponds to the length of the pipe and the bandwidth gives the diameter of the pipe, then the delay×bandwidth product gives the volume of the pipe the maximum number of bits that could be in transit through the pipe at any given instant. If latency (measured in time) corresponds to the length of the pipe, then given the width of each bit (also measured in time) you can calculate how many bits fit in the pipe. For example, a transcontinental channel with a one-way latency of 50 ms and a bandwidth of 45 Mbps is able to hold

$$50\times10-3\ \text{s}\times45\times106\ \text{bits/s}$$

$$= 2.25\times106\ \text{bits}$$

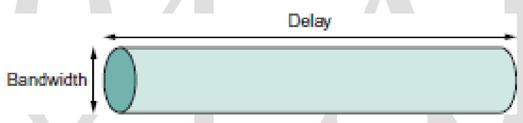
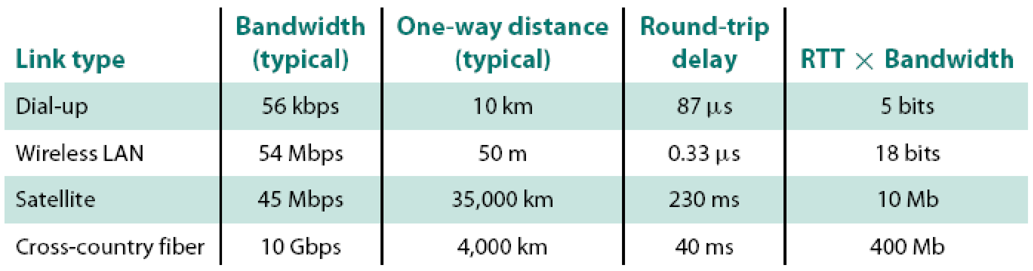


Fig: Network as a pipe

The delay×bandwidth product is important to know when constructing high-performance networks because it corresponds to how many bits the sender must transmit before the first bit arrives at the receiver. If the sender is expecting the receiver to somehow signal that bits are starting to arrive, and it takes another channel latency for this signal to propagate back to the sender, then the sender can send up one RTT×bandwidth worth of data before hearing from the receiver that all is well. The bits in the pipe are said to be –in flight,‖ which means that if the receiver tells the sender to stop transmitting it might receive up to one RTT×bandwidth‘s worth of data before the sender manages to respond.



Link type	Bandwidth (typical)	One-way distance (typical)	Round-trip delay	RTT × Bandwidth
Dial-up	56 kbps	10 km	87 μs	5 bits
Wireless LAN	54 Mbps	50 m	0.33 μs	18 bits
Satellite	45 Mbps	35,000 km	230 ms	10 Mb
Cross-country fiber	10 Gbps	4,000 km	40 ms	400 Mb

Table: Sample Delay X Bandwidth Products

High-Speed Networks

The bandwidths available on today‘s networks are increasing at a dramatic rate, and there is eternal optimism that network bandwidth will continue to improve. This causes network designers to start thinking about what happens in the limit or, stated another way, what is the impact on network design of having infinite bandwidth available.

Although high-speed networks bring a dramatic change in the bandwidth available to applications, in many respects their impact on how we think about networking comes in what does *not* change as bandwidth increases: the speed of light. In effect, the 1-MB file looks like a stream of data that needs to be transmitted across a 1-Mbps network, while it looks like a single packet on a 1-Gbps network.

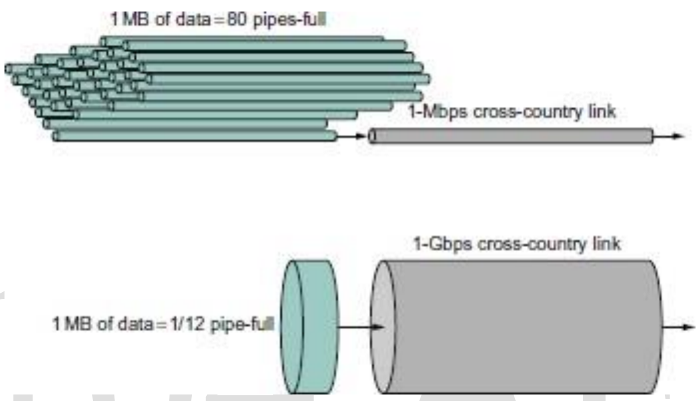

1 MB of data = 80 pipes-full
1-Mbps cross-country link
1-Gbps cross-country link
1 MB of data = 1/12 pipe-full

Fig: Relationship between bandwidth and latency. A 1-MB file would fill the 1-Mbps link 80 times but only fill the 1-Gbps link 1/12 of one time.

Perhaps the best way to understand the relationship between throughput and latency is to return to basics. The effective end-to-end throughput that can be achieved over a network is given by the simple relationship

$$\text{Throughput} = \text{TransferSize}/\text{TransferTime}$$

where TransferTime includes not only the elements of one-way Latency identified earlier in this section, but also any additional time spent requesting or setting up the transfer. Generally, we represent this relationship as

$$\text{TransferTime} = \text{RTT}+1/\text{Bandwidth}\times\text{TransferSize}$$

5 C. Discuss the IP addressing methods [MAY/JUNE-2014 , APR/MAY-2011], Nov 16

It was possible to build reasonably large LANs using bridges and LAN switches, but that such approaches were limited in their ability to scale and to handle heterogeneity. Beyond the limitations of bridged networks, it is possible to build large, highly heterogeneous networks with reasonably efficient routing, such networks as *internetworks*.

Internetwork

The term *internetwork* or sometimes just *internet* with a lowercase *i*, to refer to an arbitrary collection of networks interconnected to provide some sort of host-to-host packet delivery service. N*etwork is* to mean either a directly connected or a switched network of the kind described in the previous section and the previous chapter. Such a network uses one technology, such as 802.11 or Ethernet. An *inter- network* is an interconnected collection of such networks. Sometimes, to avoid ambiguity, we refer to the underlying networks that we are interconnecting as *physical* networks. An internet is a *logical* network built out of a collection of physical networks. In this context, a collection of Ethernets connected by bridges or switches would still be viewed as a single network.

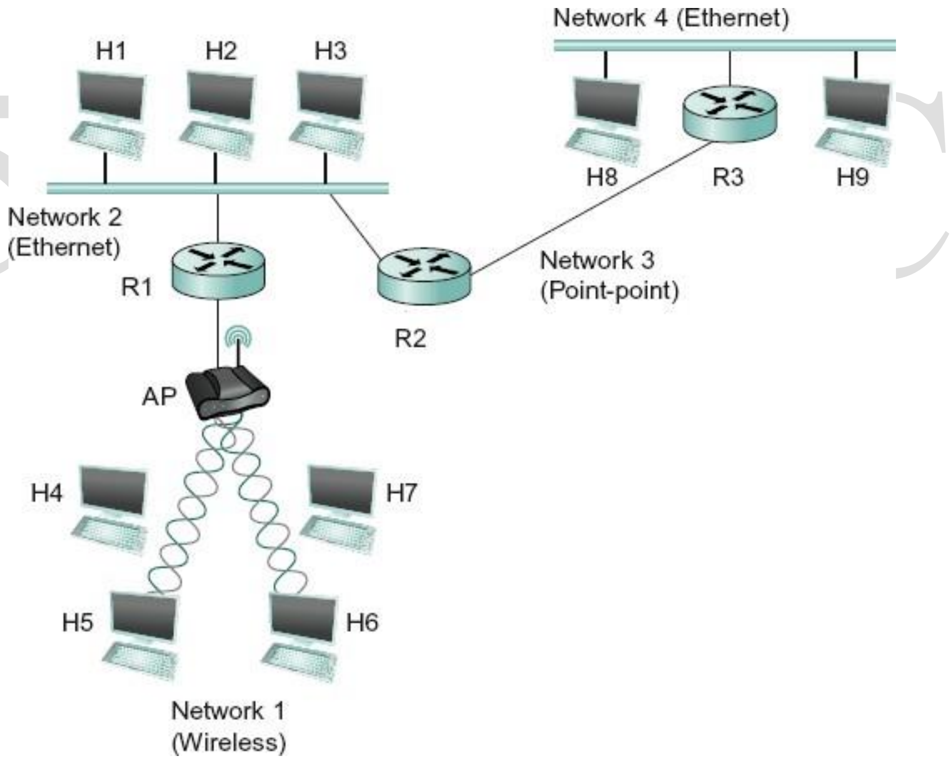


Fig: A simple internetwork. Hn =host; Rn =router.

For example, the below figure shows how hosts H5 and H8 are logically connected by the internet in the above figure, including the protocol graph running on each node.

The figure above shows an example internetwork. An internetwork is often referred to as a –network of networks‖ because it is made up of lots of smaller networks. In this figure, we see Ethernets, a wireless network, and a point-to-point link. Each of these is a single-technology network. The nodes that interconnect the networks are called *routers*. They are also sometimes called *gateways*, but since this term has several other intentions.

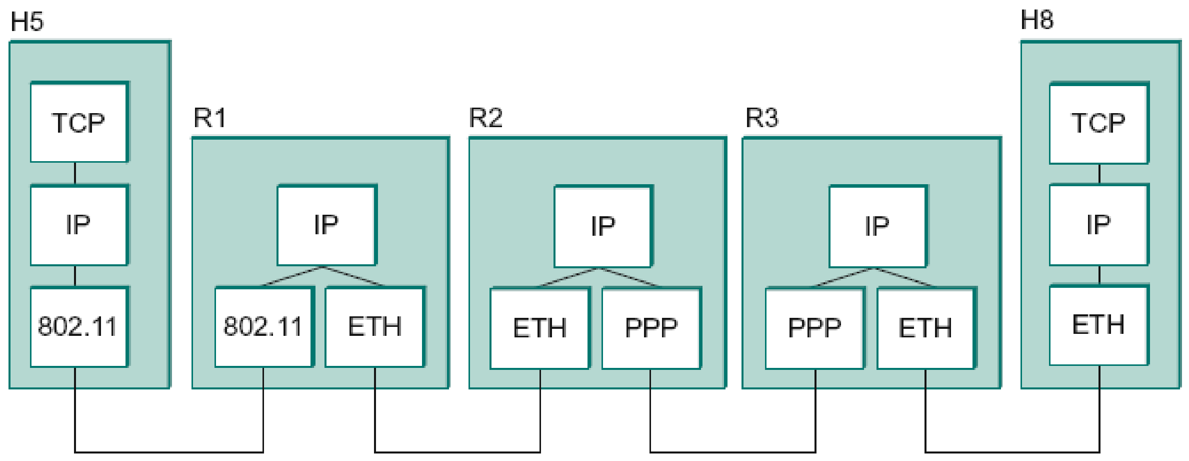


Fig: A simple internetwork, showing the protocol layers used to connect H5 to H8. ETH is the protocol that runs over the Ethernet.

The *Internet Protocol* is the key tool used today to build scalable, heterogeneous internetworks. It was originally known as the Kahn-Cerf protocol after its inventors. One way to think of IP is that it runs on all the nodes (both hosts and routers) in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork.

Service Model

When you build an internetwork is to define its *service model*, that is, the host-to-host services you want to provide. The main concern in defining a service model for an internetwork is that we can provide a host-to-host service only if this service can somehow be provided over each of the underlying physical networks.

For example, it would be no good deciding that our internetwork service model was going to provide guaranteed delivery of every packet in 1 ms or less if there were underlying network technologies that could arbitrarily delay packets.

The IP service model can be thought of as having two parts: an addressing scheme, which provides a way to identify all hosts in the internetwork, and a datagram (connectionless)model of data delivery. This service model is sometimes called *best effort* because, although IP makes every effort to deliver datagrams, it makes no guarantees.

The IP datagram is fundamental to the Internet Protocol. A datagram is a type of packet that happens to be sent in a connectionless manner over a network. Every datagram carries enough information to let the network forward the packet to its correct destination; there is no need for any advance setup mechanism to tell the network what to do when the packet arrives. Just send the information, and the network makes its best effort to get it to the desired destination.

The ―best-effort‖ part means that if something goes wrong and the packet gets lost, corrupted, misdelivered, or in any way fails to reach its intended destination, the network does nothing—it made its best effort, and that is all it has to do. It does not make any attempt to recover from the failure. This is sometimes called an *unreliable* service.

Best-effort delivery does not just mean that packets can get lost. Sometimes they can get delivered out of order, and sometimes the same packet can get delivered more than once. The higher-level protocols or applications that run above IP need to be aware of all these possible failure modes.

Packet Format

The key part of the IP service model is the type of packets that can be carried. The IP datagram, like most packets, consists of a header followed by a number of bytes of data. The format of the header is shown in the figure below.

Looking at each field in the IP header, the –simple‖ model of best-effort datagram delivery still has some subtle features. The Version field specifies the version of IP. The current version of IP is 4, and it is sometimes called *IPv4*.

The next field, HLen, specifies the length of the header in 32-bit words. When there are no options, which is most of the time, the header is 5 words (20 bytes) long. The 8-bit TOS (type of service) field has had a number of different definitions over the years, but its basic function is to allow packets to be treated differently based on application needs.

For example, the TOS value might determine whether or not a packet should be placed in a special queue that receives low delay. The next 16 bits of the header contain the Length of the datagram, including the header. Unlike the HLen field, the Length field counts bytes rather than words. Thus, the maximum size of an IP datagram is 65,535 bytes.

The physical network, over which IP is running, however, may not support such long packets. For this reason, IP supports a fragmentation and reassembly process.

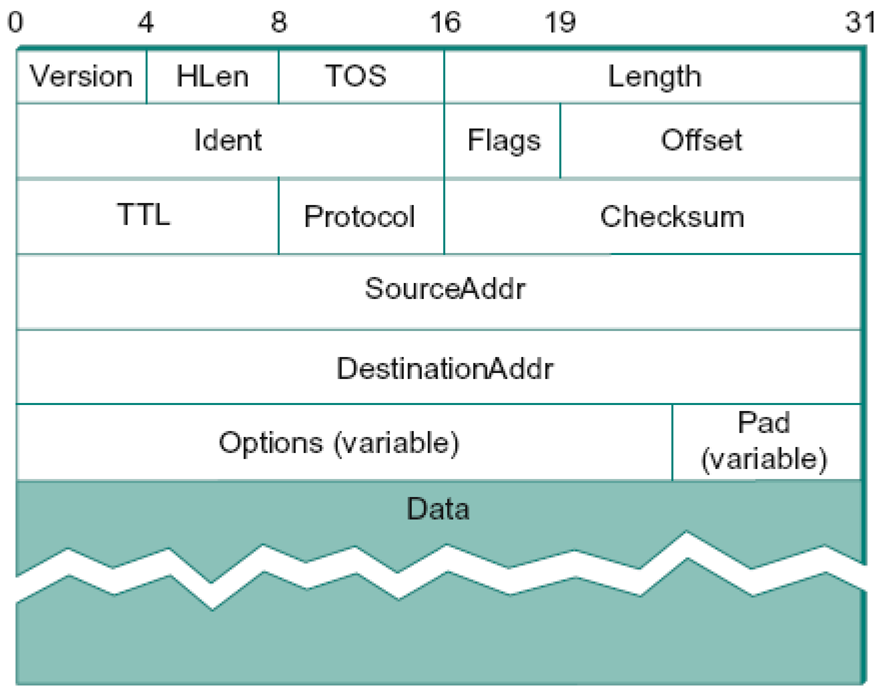


Fig: IPv4 packet header.

The second word of the header contains information about fragmentation. Moving on to the third word of the header, the next byte is the TTL (time to live) field. Its name reflects its historical meaning rather than the way it is commonly used today. The intent of the field is to catch packets that have been going around in routing loops and discard them, rather than let them consume resources indefinitely.

The Checksum is calculated by considering the entire IP header as a sequence of 16-bit words, adding them up using ones complement arithmetic, and taking the ones complement of the result. The last two required fields in the header are the SourceAddr and the DestinationAddr for the packet. Every packet contains a full address for its intended destination so that forwarding decisions can be made at each router. The source address is required to allow recipients to decide if they want to accept the packet and to enable them to reply.

Fragmentation and Reassembly

One of the problems of providing a uniform host-to-host service model over a heterogeneous collection of networks is that each network technology tends to have its own idea of how large a packet can be. For example, an Ethernet can accept packets up to 1500 bytes long, while FDDI (Fiber Distributed Data Interface) packets may be 4500 bytes long.

This leaves two choices for the IP service model: Make sure that all IP datagrams are small enough to fit inside one packet on any network technology, or provide a means by which packets can be fragmented and reassembled when they are too big to go over a given network technology.

The central idea here is that every network type has a *maximum transmission unit* (MTU), which is the largest IP datagram that it can carry in a frame. This value is smaller than the largest packet size on that network because the IP datagram needs to fit in the *payload* of the link-layer frame.8

When a host sends an IP datagram, therefore, it can choose any size that it wants. A reasonable choice is the MTU of the network to which the host is directly attached. Then, fragmentation will only be necessary if the path to the destination includes a network with a smaller MTU. Should the transport protocol that sits on top of IP give IP a packet larger than the local MTU, however, then the source host must fragment it.

Fragmentation typically occurs in a router when it receives a datagram that it wants to forward over a network that has an MTU that is smaller than the received datagram. To enable these fragments to be reassembled at the receiving host, they all carry the same identifier in the Ident field.

This identifier is chosen by the sending host and is intended to be unique among all the datagrams that might arrive at the destination from this source over some reasonable time period. Since all fragments of the original datagram contain this identifier, the reassembling host will be able to recognize those fragments that go together. Should all the fragments not arrive at the receiving host, the host gives up on the reassembly process and discards the fragments that did arrive. IP does not attempt to recover from missing fragments.

For example, consider what happens when host H5 sends a datagram to host H8 Assuming that the MTU is 1500 bytes for the two Ethernets and the 802.11 network, and 532 bytes for the point-to-point network, then a 1420-byte datagram (20-byte IP header plus 1400 bytes of data) sent from H5 makes it across the 802.11 network and the first Ethernet without fragmentation but must be fragmented into three datagrams at router R2.

These three fragments are then forwarded by router R3 across the second Ethernet to the destination host. This situation is illustrated in the figure below. This figure also serves to reinforce two important points:

- Each fragment is itself a self-contained IP datagram that is transmitted over a sequence of physical networks, independent of the other fragments.
- Each IP datagram is re-encapsulated for each physical network over which it travels.

The fragmentation process can be understood in detail by looking at the header fields of each datagram. The unfragmented packet, shown at the top, has 1400 bytes of data and a 20-byte IP header. When the packet arrives at router R2, which has an MTU of 532 bytes, it has to be fragmented. A 532-byte MTU leaves 512 bytes for data after the 20-byte IP header, so the first fragment contains 512 bytes of data.
The router sets the Mbit in the Flags field meaning that there are more fragments to follow, and it sets the Offset to 0, since this fragment contains the first part of the original datagram.

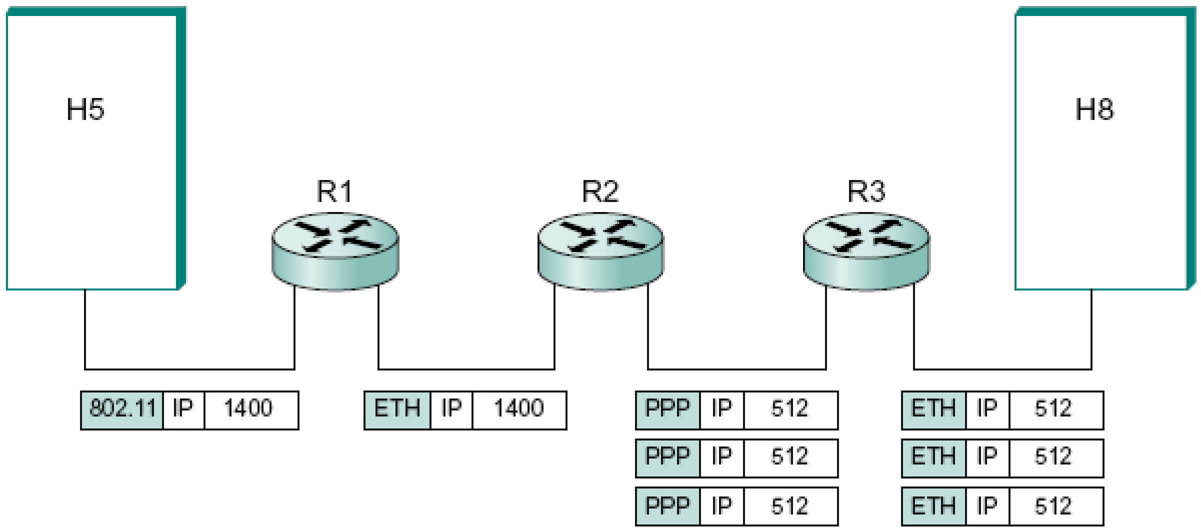


Fig: IP datagrams traversing the sequence of physical networks

The data carried in the second fragment starts with the 513th byte of the original data, so the Offset field in this header is set to 64, which is 512÷8. Why the division by 8? Because the designers of IP decided that fragmentation should always happen on 8-byte boundaries, which means that the Offset field counts 8-byte chunks, not bytes.

The third fragment contains the last 376 bytes of data, and the offset is now 2×512÷8 = 128. Since this is the last fragment, the Mbit is not set. Observe that the fragmentation process is done in such a way that it could be repeated if a fragment arrived at another network with an even smaller MTU.

Fragmentation produces smaller, valid IP datagrams that can be readily reassembled into the original datagram upon receipt, independent of the order of their arrival.

Reassembly is done at the receiving host and not at each router. IP reassembly is far from a simple process. For example, if a single fragment is lost, the receiver will still attempt to reassemble the datagram, and it will eventually give up and have to garbage-collect the resources that were used to perform the failed reassembly. For this reason, among others, IP fragmentation is generally considered a good thing to avoid.

Hosts are now strongly encouraged to perform ―path MTU discovery,‖ a process by which fragmentation is avoided by sending packets that are small enough to traverse the link with the smallestMTUin the path fromsender to receiver.

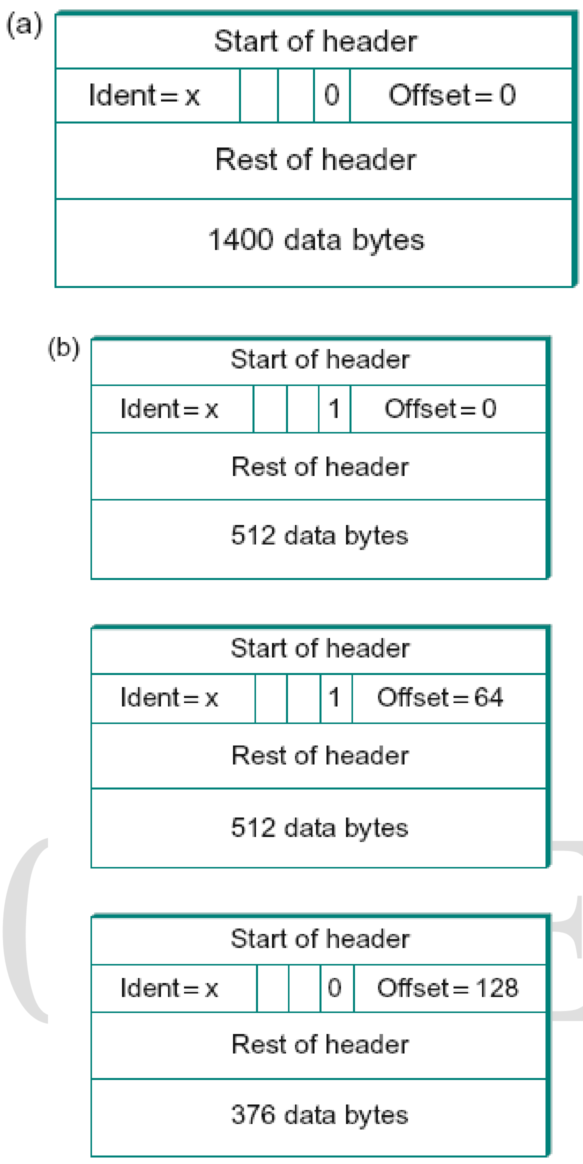


Fig: Header fields used in IP fragmentation: (a) unfragmented packet; (b) fragmented packet

Global Addresses

A global addressing scheme is one in which no two hosts have the same address. Global uniqueness is the first property that should be provided in an addressing scheme. Ethernet addresses are globally unique, but that alone does not suffice for an addressing scheme in a large internetwork.

Ethernet addresses are also *flat*, which means that they have no structure and provide very few clues to routing protocols. In contrast, IP addresses are *hierarchical*, by which it means that they are made up of several parts that correspond to some sort of hierarchy in the internetwork. Specifically, IP addresses consist of two parts, usually referred to as a *network* part and a *host* part. This is a fairly logical structure for an internetwork, which is made up of many interconnected networks.

The network part of an IP address identifies the network to which the host is attached; all hosts attached to the same network have the same network part in their IP address.

The host part then identifies each host uniquely on that particular network. Originally, IP addresses were divided into three different classes, as shown in the figure below, each of which defines different-sized network and host parts. In all cases, the address is 32 bits long.

The class of an IP address is identified in the most significant few bits. If the first bit is 0, it is a class A address. If the first bit is 1 and the second is 0, it is a class B address. If the first two bits are 1 and the third is 0, it is a class C address. Thus, of the approximately 4 billion possible IP addresses, half are class A, one-quarter are class B, and one-eighth are class C. Each class allocates a certain number of bits for the network part of the address and the rest for the host part.

Class A networks have 7 bits for the network part and 24 bits for the host part, meaning that there can be only 126 class A networks (the values 0 and 127 are reserved), but each of them can accommodate up to 224 −2 (about 16 million) hosts (again, there are two reserved values). Class B addresses allocate 14 bits for the network and 16 bits for the host, meaning that each class B network has room for 65,534 hosts.

Finally, class C addresses have only 8 bits for the host and 21 for the network part. Therefore, a class C network can have only 256 unique host identifiers, which means only 254 attached hosts. However, the addressing scheme supports 221 class C networks.

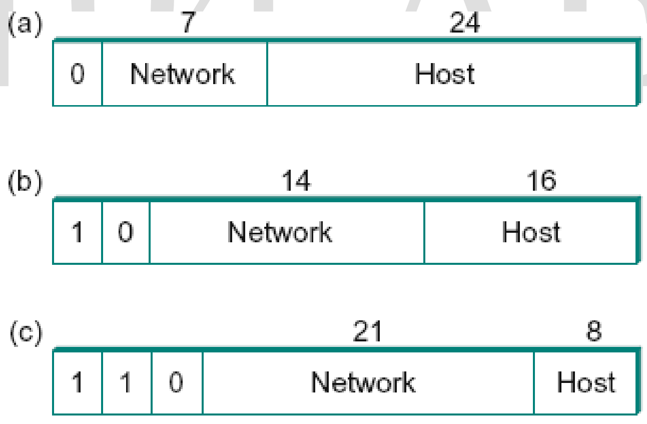


Fig: IP addresses: (a) class A; (b) class B; (c) class C.

Datagram Forwarding in IP

Every IP datagram contains the IP address of the destination host. The network part of an IP address uniquely identifies a single physical network that is part of the larger Internet.

All hosts and routers that share the same network part of their address are connected to the same physical network and can thus communicate with each other by sending frames over that network.

Every physical network that is part of the Internet has at least one router that, by definition, is also connected to at least one other physical network; this router can exchange packets with hosts or routers on either network. Forwarding IP datagrams can therefore be handled in the following way.

A datagram is sent from a source host to a destination host, possibly passing through several routers along the way. Any node, whether it is a host or a router, first tries to establish whether it is connected to the same physical network as the destination. To do this, it compares the network part of the destination address with the network part of the address of each of its network interfaces. If a match occurs, then that means that the destination lies on the same physical network as the interface, and the packet can be directly delivered over that network.

The datagram forwarding algorithm:

```
if (NetworkNum of destination = NetworkNum of one of my interfaces) then
deliver packet to destination over that interface
else
if (NetworkNum of destination is in my forwarding table) then
deliver packet to NextHop router
else
deliver packet to default router
```

For a host with only one interface and only a default router in its forwarding table, this simplifies to

```
if (NetworkNum of destination = my NetworkNum) then
deliver packet to destination directly
else
deliver packet to default router
```

Subnetting

Subnetting provides a first step to reducing total number of network numbers that are assigned. The idea is to take a single IP network number and allocate the IP addresses with that network number to several physical networks, which are now referred to as *subnets*.

Several things need to be done to make this work. First, the subnets should be close to each other. This is because at a distant point in the Internet, they will all look like a single network, having only one network number between them. This means that a router will only be able to select one route to reach any of the subnets, so they had better all be in the same general direction.

A perfect situation in which to use subnetting is a large campus or corporation that has many physical networks. From outside the campus, all you need to know to reach any subnet inside the campus is where the campus connects to the rest of the Internet. This is often at a single point, so one entry in your forwarding table will suffice. Even if there are multiple points at which the campus is connected to the rest of the Internet, knowing how to get to one point in the campus network is still a good start.

The mechanism by which a single network number can be shared among multiple networks involves configuring all the nodes on each subnet with a *subnet mask*. With simple IP addresses, all hosts on the same network must have the same network number. The subnet mask enables us to introduce a *subnet number*; all hosts on the same physical network will have the same subnet number, which means that hosts may be on different physical networks but share a single network number. This concept is illustrated in the figure below.

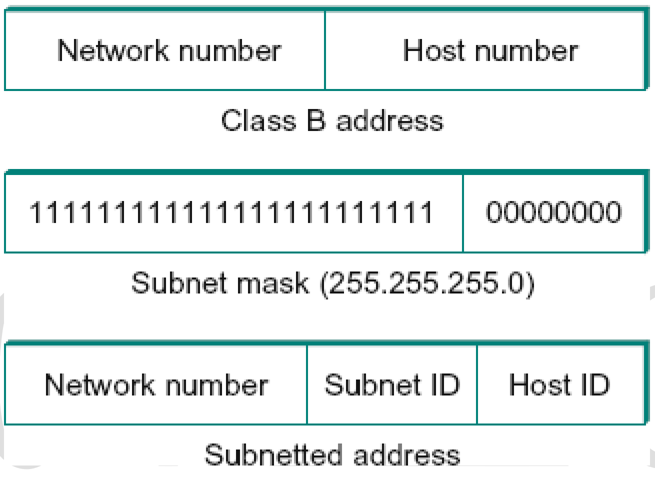


Fig: Subnet addressing

For example, host H1 in the below figure is configured with an address of 128.96.34.15 and a subnet mask of 255.255.255.128. The bitwise AND of these two numbers defines the subnet number of the host and of all other hosts on the same subnet. In this case, 128.96.34.15 AND 255.255.255.128 equals 128.96.34.0, so this is the subnet number for the topmost subnet in the figure. When the host wants to send a packet to a certain IP address, the first thing it does is to perform a bitwise AND between its own subnet mask and the destination IP address. If the result equals the subnet number of the sending host, then it knows that the destination host is on the same subnet and the packet can be delivered directly over the subnet.

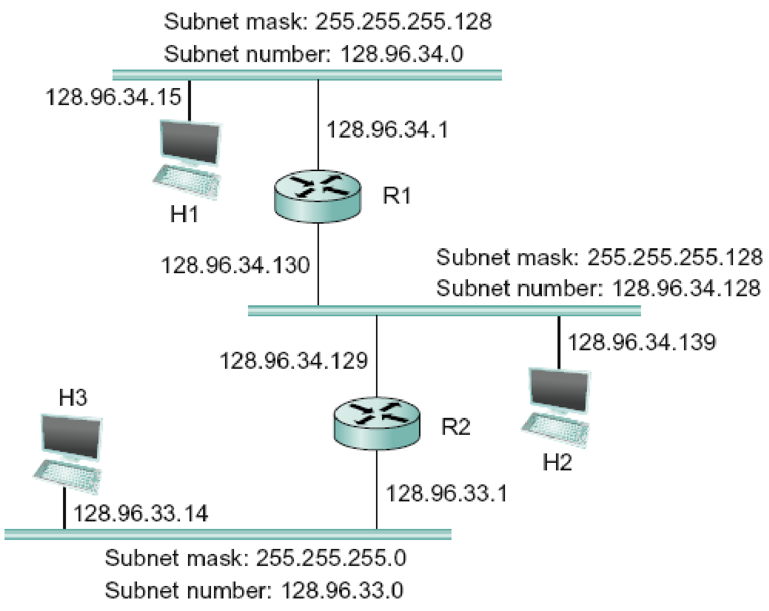
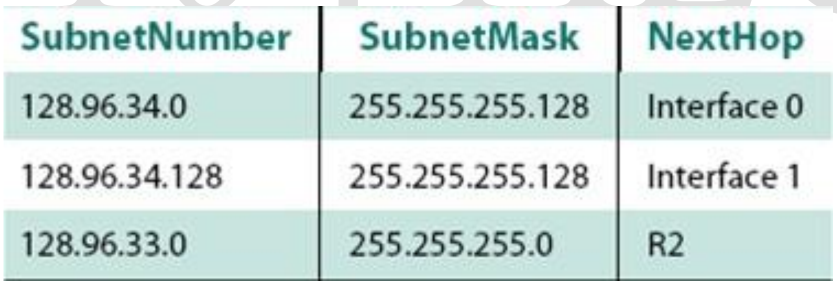

Subnet mask: 255.255.255.128
Subnet number: 128.96.34.0
128.96.34.15
128.96.34.1
H1
R1
128.96.34.130
Subnet mask: 255.255.255.128
Subnet number: 128.96.34.128
128.96.34.139
128.96.34.129
H3
R2
H2
128.96.33.1
128.96.33.14
Subnet mask: 255.255.255.0
Subnet number: 128.96.33.0

Fig: An example of subnetting

If the results are not equal, the packet needs to be sent to a router to be forwarded to another subnet. For example, if H1 is sending to H2, then H1 ANDs its subnet mask (255.255.255.128) with the address for H2 (128.96.34.139) to obtain 128.96.34.128. This does not match the subnet number for H1 (128.96.34.0) so H1 knows that H2 is on a different subnet. Since H1 cannot deliver the packet to H2 directly over the subnet, it sends the packet to its default router R1.



SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Fig: Forwarding Table with Subnetting

INDUSTRY CONNECTIVITY AND LATEST DEVELOPMENTS

Industry Connectivity:-

The following companies (Industries) are connectivity to computer Latest Developments.

- LIFI if fast data transfer in networking.
- New revolution in 5G, 6G in all the smart phone, smart televisions, etc.
- Network on chip
- LTE were introduced almost in all the home appliances