INTERNET TECHNOLOGY UNIT 1,2,3,4,5,6

TYBSC(IT) SEM 6

COMPILED BY : SIDDHESH ZELE

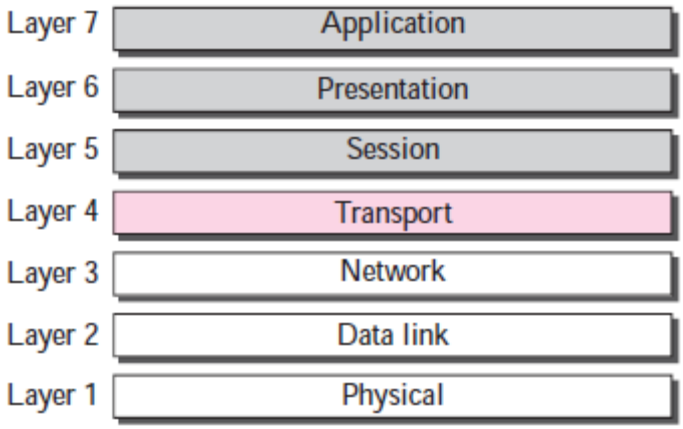
302 PARANJPE UDYOG BHAVAN, NEAR KHANDELWAL SWEETS, NEAR THANE STATION , THANE (WEST)

PHONE NO: 8097071144 / 8097071155 / 8655081002

Syllabus

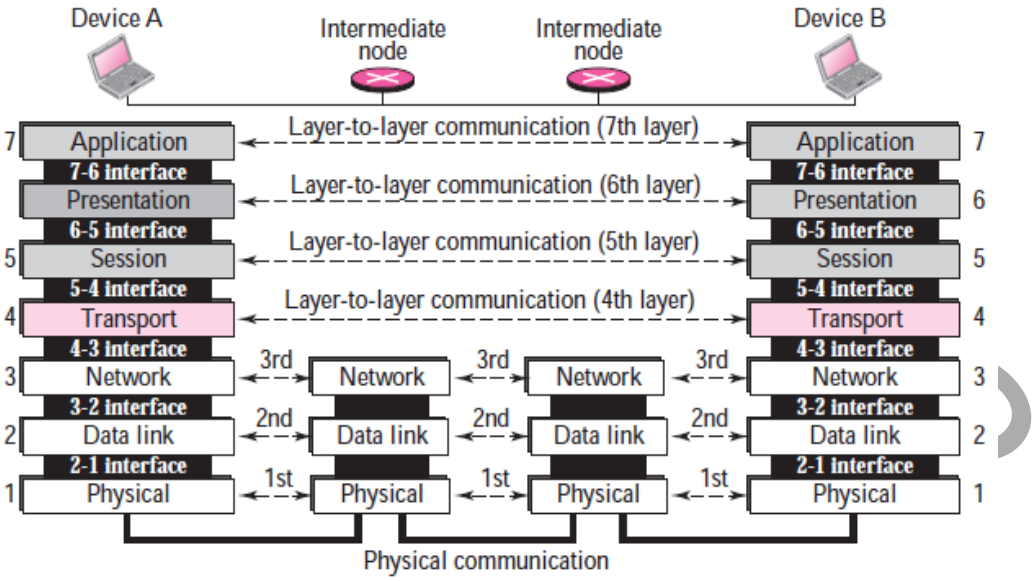
UNIT 1 INTRODUCTION

- The **International Standards Organization (ISO)** is a multinational body dedicated to worldwide agreement on international standards. Almost three-fourths of countries in the world are represented in the ISO. An ISO standard that covers all aspects of network communications is the Open Systems Interconnection(OSI) model.
- It was first introduced in the late 1970s. An open system is a set of protocols that allows any two different systems to communicate.
- The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems.
- It consists of seven separate but related layers each of which defines a part of the process of moving information across a network.



Layered Architecture –

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



Layer-to-Layer Communication

- In Figure – device A sends a message to device B (through intermediate nodes). At the sending site, the message is moved down from layer 7 to layer 1. At layer 1 the entire package is converted to a form that can be transferred to the receiving site. At the receiving site, the message is moved up from layer 1 to layer 7.

1. **Interfaces between Layers**
   The passing of the data and network information down through the layers of the sending device and back up through the layers of the receiving device is made possible by an **interface** between each pair of adjacent layers. Each interface defines what information and services a layer must provide for the layer above it.
2. **Organization of the Layers**
   The seven layers can be thought of as belonging to three subgroups. Layers 1, 2, and 3 physical, data link, and network—are the network support layers; they deal with the physical aspects of moving data from one device to another (such as electrical specifications, physical connections, physical addressing, and transport timing and reliability). Layers 5, 6, and 7 session, presentation, and application—can be thought of as the user support layers; they allow interoperability among unrelated software systems. Layer 4, the transport layer, links the two subgroups and ensures that what the lower layers have transmitted is in a form that the upper layers can use.

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

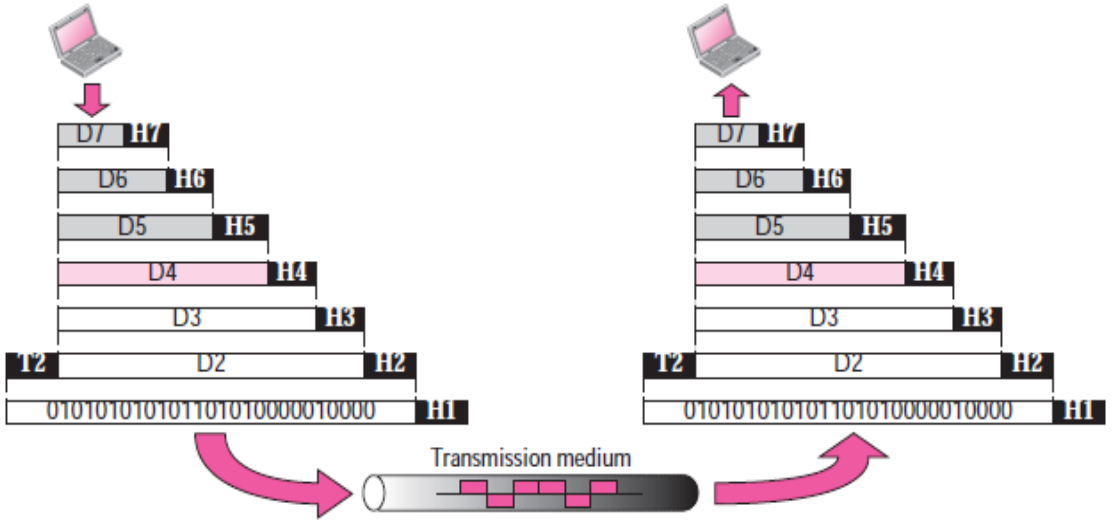
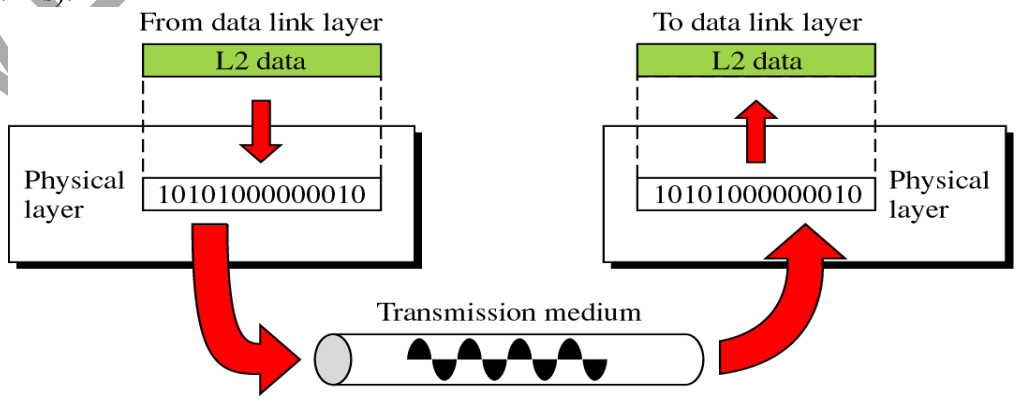


Figure – An exchange using OSI model

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

Physical Layer –

- The physical layer is responsible for the movement of individual bits from one node to another.
- The physical layer coordinates the function required to carry a bit stream over a physical medium.
- It deals with the mechanical and electrical specifications of the interface and transmission medium.
- It also defines the procedures and functions that physical devices and the interfaces have to perform for transmission to occur. The data here consists of a stream of bits (sequence of 0's and 1's).

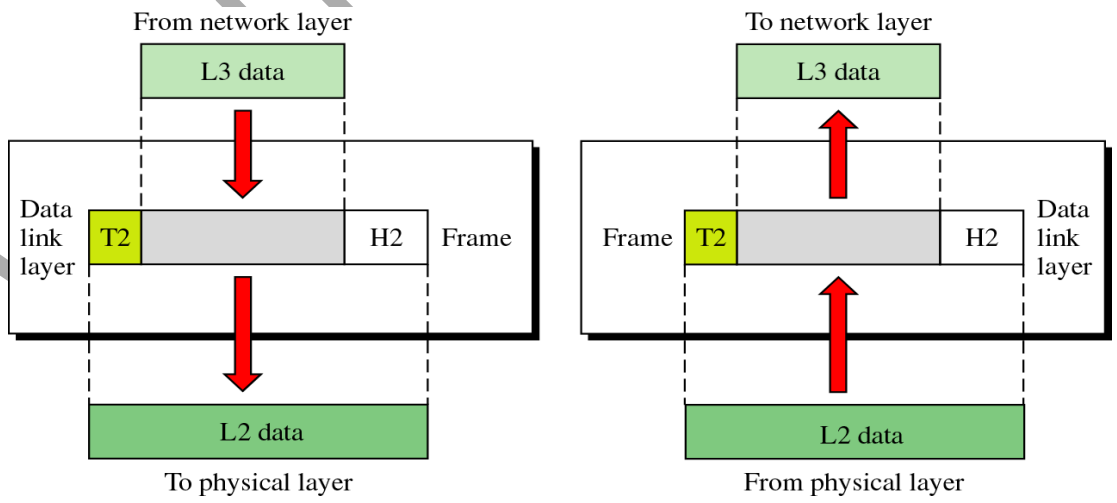


Responsibilities of Physical Layer –

1. **Physical characteristics of interfaces and medium**: It defines the characteristics of the interface between the devices and the transmission medium.
2. **Representation of bits**: It consist of stream of bits (0;s and 1's) with no interpretation. The bits are represented in the form of signals i.e. in the encoded format.
3. **Data Rate**: The transmission rate i.e. the number of bits sent each second is also defined.
4. **Synchronization of bits**: The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.
5. **Line Configuration**: The physical Layer is concerned with the connection of devices to the media ie point to point and multi point configuration.
6. **Physical topology:** It defines how devices are connected to make a network. Devices can be connected by using a mesh topology, a star topology or any such kind of topologies.
7. **Transmission Mode**: It also defines the direction of transmission between devices : simplex, half duplex or full duplex.

Data Link Layer –

- The data link layer is responsible for moving frames from one node to the other. The data link layer divides the stream of bits received from the network layer into manageable data units called frames.
- The data link layer makes the physical layer reliable by adding mechanisms to detect and retransmit damaged or lost frames. It also controls the flow (i.e. the rate at which the data is sent or received) of data.

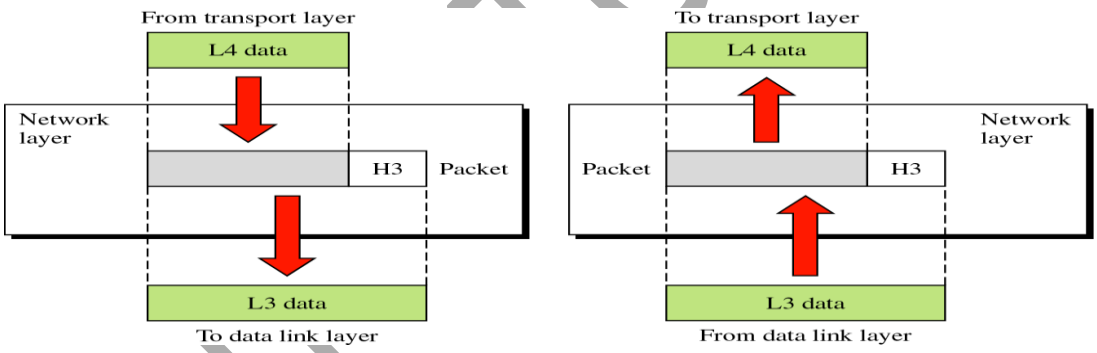


Responsibilities –

1. **Framing** The data link layer divides the stream of bits received from the network layer into manageable data units called **frames**.
2. **Physical addressing** If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender of receiver of the frame.
3. **Flow control** If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
4. **Error control** The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames.It also uses a mechanism to recognize duplicate frames
5. **Access Control** When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over link at any time.

Network Layer :

- The network layer is responsible for the source to destination delivery of a packet, possibly across multiple networks (links).
- When the links are connected to the large network than there is a need to route a packet. This routing process is done by routers which exist at the network layer.



Responsibilities –

1. **logical addressing** – The physical addressing implemented by the data link layer handles the addressing problem locally. If a packet passes the network boundary we need another addressing system to help distinguish the source and Destination systems. The network layer adds a header to the packet coming from the upper layer that among other things, includes the logical address of the sender and receiver.
2. **Routing** – When independent networks or links are connected to create internetworks or large network , the connecting devices route or switch the packets to their final destination.

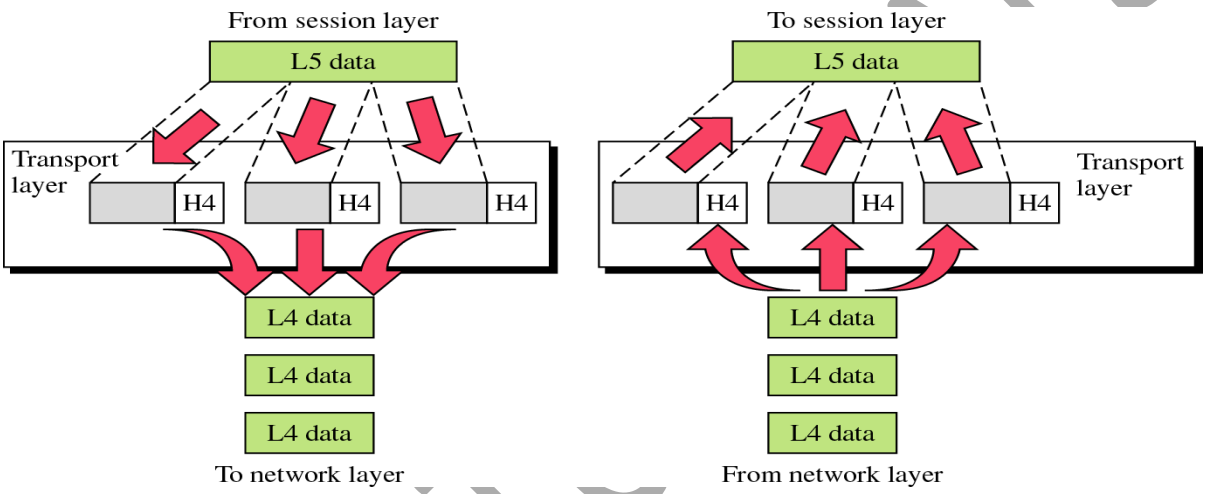
Transport Layer –

- The transport layer is responsible for process to process delivery of the entire message. A process is an application program running on a host. Like the data link layer, the transport layer is responsible for error control and flow control.

- A message is divided into segments, than it is the responsibility of the transport layer to reassemble the segments at the destination point and identify and replace the segments that were lost in transmission. This layer can either be connectionless or connection oriented.

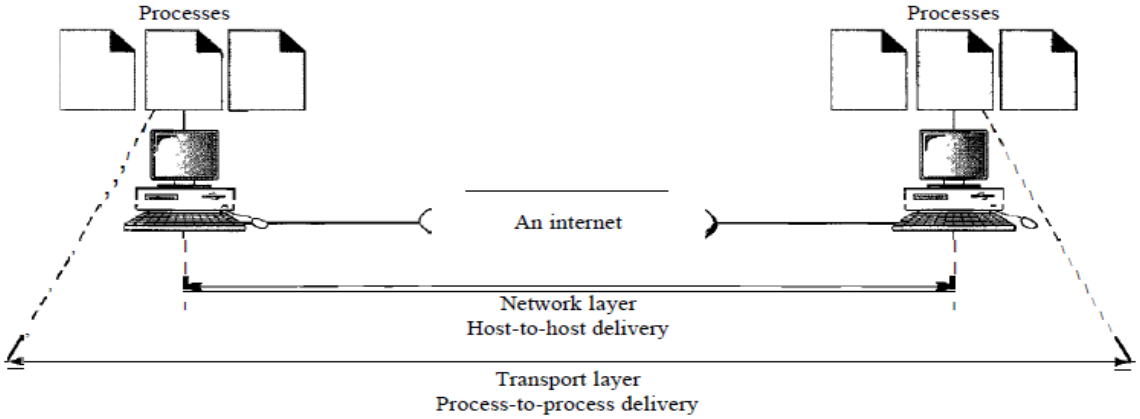
Responsibilities –

1. **Service point addressing** – The transport layer header must include a type of address called service point address to check the network layer gets each packet to the correct computer and the transport layer gets the entire message to the correct process on that computer.



2. **Segmentation and reassembly** – A message is divided into transmittable segments with each segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.
3. **Flow Control** – Like the data link layer , the transport layer is responsible for flow control. However,flow control at this layer is performed end to end rather than across a single link.
4. **Error Control** – Here the sending transport layer makes sure that the entire message arrives at the receiving transport layer without error. Error correction is usually achieved through retransmission.
5. **Connection control** – The transport layer can be either connectionless or connection oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.

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

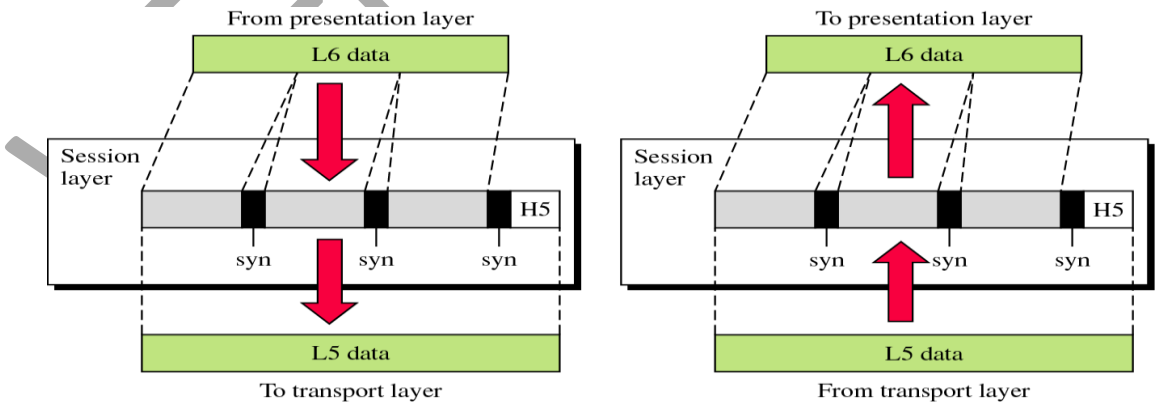


Session Layer:

- The services provided by the first 3 layers (physical, data link, network) are sometimes not sufficient for some processes. At this point, the session layer acts as a dialog controller in the network.
- It establishes, maintains and synchronizes the interaction among communicating systems. The session layer allows a process to add checkpoints to a stream of data.

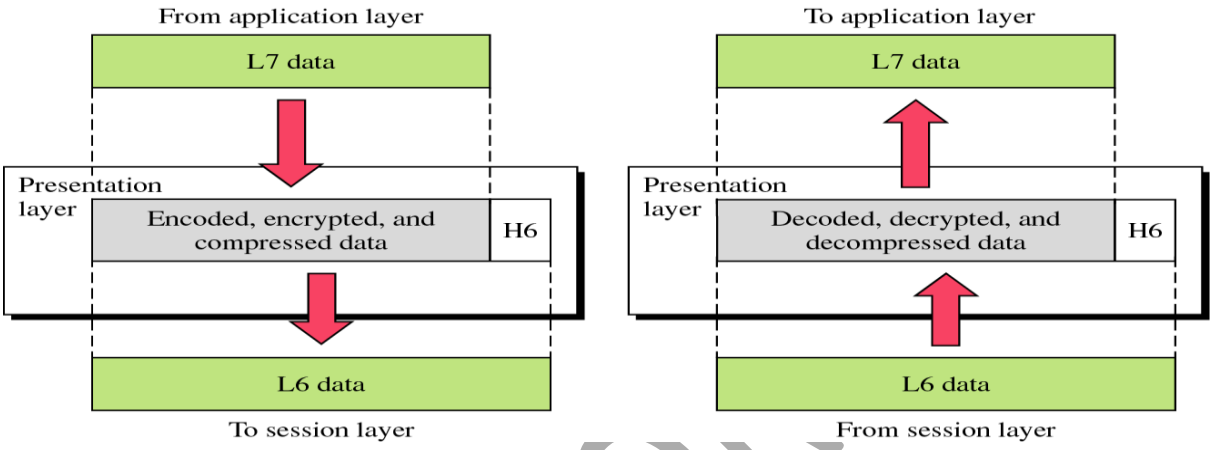
Responsibilities –

1. **Dialog control –** The session layer allows two systems to enter into dialog. It allows the communication between two processes to take place in either half duplex or full duplex.
2. **Synchronization –** The session Layer allows a process to add checkpoints or synchronization points to a stream of data.
   Figure – illustrates the relationship of the session layer to the transport and presentation layers.



Presentation Layer:

- The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.
- The presentation layer is responsible for translation, compression and encryption.

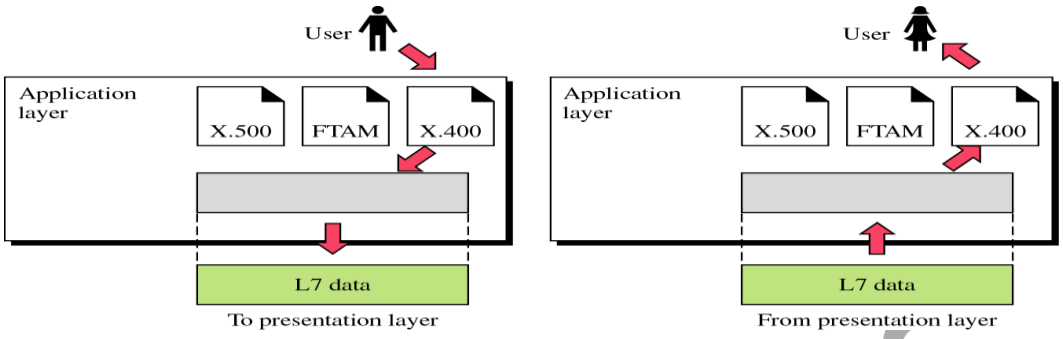


Responsibilities –

1. **Translation** – The processes in two systems are usually exchanging information in the form of character strings, numbers and so on.The information must be changed to bit streams before being transmitted.
2. **Encryption –** To carry sensitive data, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to another from and sends the resulting message out over the network. Decryption reverses the original process.
3. **Compression –** Data compression reduces the number of bits contained in the information.

Application Layer:

- The application layer enables the user, whether human or software, to access the network.
- It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services.
- Figure – shows the relationship of the application layer to the user and the presentation layer. Of the many application services available, the figure shows only three: *XAOO* (message-handling services), X.500 (directory services), and file transfer, access, and management (FTAM).
- The user in this example employs *XAOO* to send an e-mail message.



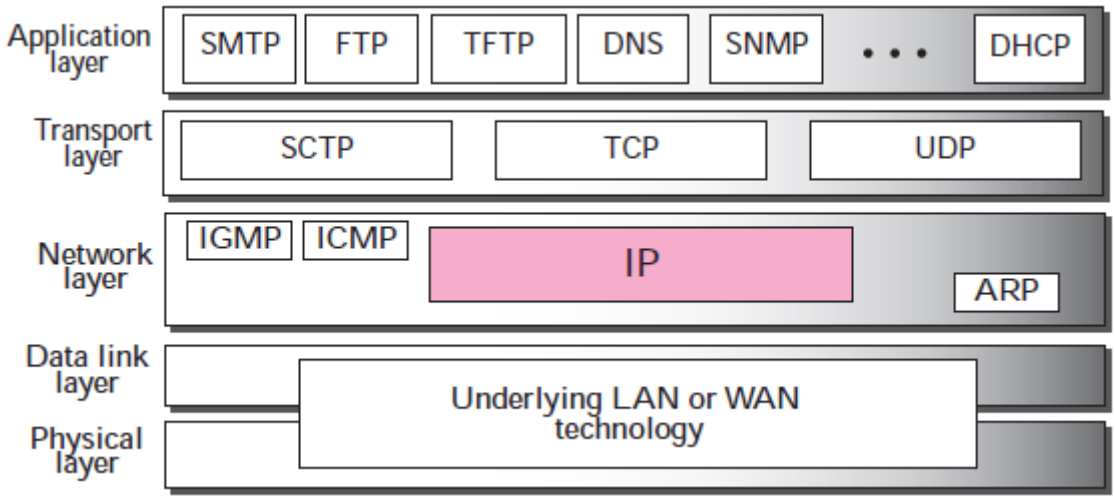
The application layer is responsible for providing services to the user.

1. **Network virtual Terminal –** It is a software version of a physical terminal, and it allows a user to log on to a remote host. To do so the application creates a software emulation of a terminal at the remote host.The user computer talks to the software emulation of a terminal which in turn talks to the host and vice versa.
2. **File transfer, access a nd management –** The application allowas a user to access files in a remote host, to retrieve files from a remote computer for use in the local computer, and to manage a control files in a remote computer locally.
3. **Mail service –** The application provides the basis for email forwarding and storage.
4. **Directory services –** This application provides distributed database sources and access for global information about various objects and services.

TCP/IP Protocol Suite –

Explain the Layers of TCP/IP Model –

- The **TCP/IP protocol suite** was developed prior to the OSI model. Therefore, the layers in the TCP/IP protocol suite do not match exactly with those in the OSI model. The original TCP/IP protocol suite was defined as four software layers built upon the hardware.
- TCP/IP is a **hierarchical** protocol made up of **interactive modules.** The ***modules*** are not necessarily interdependent. The term ***hierarchical*** means that each upper-level protocol is supported by one or more lower level protocols.



Physical Layers:

- In Physical Layer, TCP/IP does not define any specific protocol. It supports all the standard and proprietary protocols. A network in a TCP/IP internetwork can be local-area network or wide-area network.
- At this level, the communication is between two hops or nodes, either a computer or router. The unit of communication is a single bit. When the connection is established between the two nodes, a stream of bits is flowing between them. The physical layer, however, treats each bit individually.
- if a node is connected to *n* links, it needs *n* physical-layer protocols, one for each link. The reason is that different links may use different physical-layer protocols. The figure, however, shows only physical layers involved in the communication. Each computer involves with only one link; each router involves with only two links.
- As Figure shows, the journey of bits between computer A and computer B is made of four independent short trips. Computer A sends each bit to router R1 in the format of the protocol used by link 1. Router 1 sends each bit to router R3 in the format dictated by the protocol used by link 3. And so on.
- Router R1 has two three physical layers The layer connected to link 1 receives bits according to the format of the protocol used by link 1; the layer connected to link 3 sends bits according to the format of the protocol used by link 3. It is the same situation with the other two routers involved in the communication.

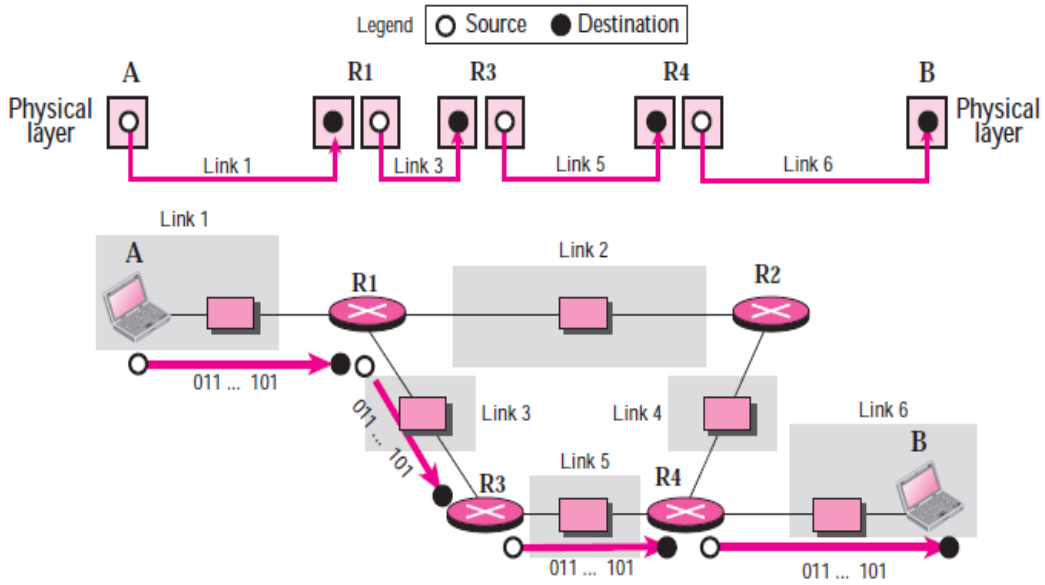


Figure – communication at physical layer

Data Link Layer –

- The unit of communication however, is a packet called a **frame**. A frame is a packet that encapsulates the data received from the network layer with an added header and sometimes a trailer. Figure shows the communication at the data link layer.
- Note that the frame that is travelling between computer A and router R1 may be different from the one travelling between router R1 and R3. When the frame is received by router R1, this router passes the frame to the data link layer protocol shown at the left. The frame is opened, the data are removed. The data are then passed to the data link layer protocol shown at the right to create a new frame to be sent to the router R3.
- The reason is that the two links, link 1 and link 3, may be using different protocols and require frames of different formats. Note also that the figure does not show the physical movement of frames; the physical movement happens only at the physical layer. The two nodes communicate logically at the data link layer, not physically.
- In other words, the data link layer at router R1 only *thinks* that a frame has been sent directly from the data link layer at computer A. What is sent from A to R1 is a stream of bits from one physical layer to another. Since a frame at A is transformed to a stream of bits, and the bits at R1 are transformed to a frame, it gives this impression to the two data link layer that a frame has been exchanged.

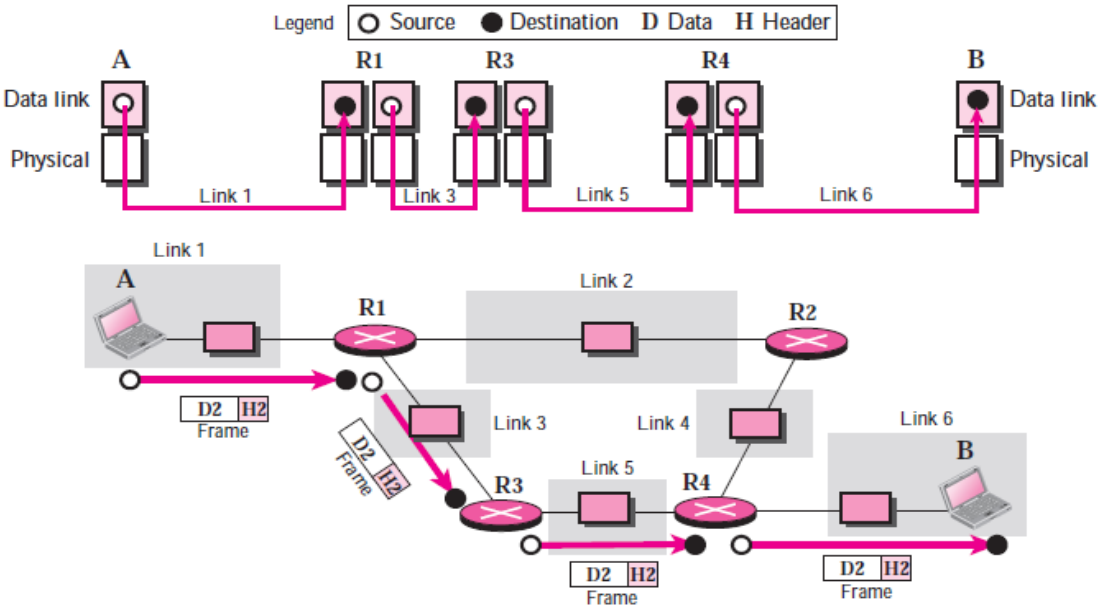
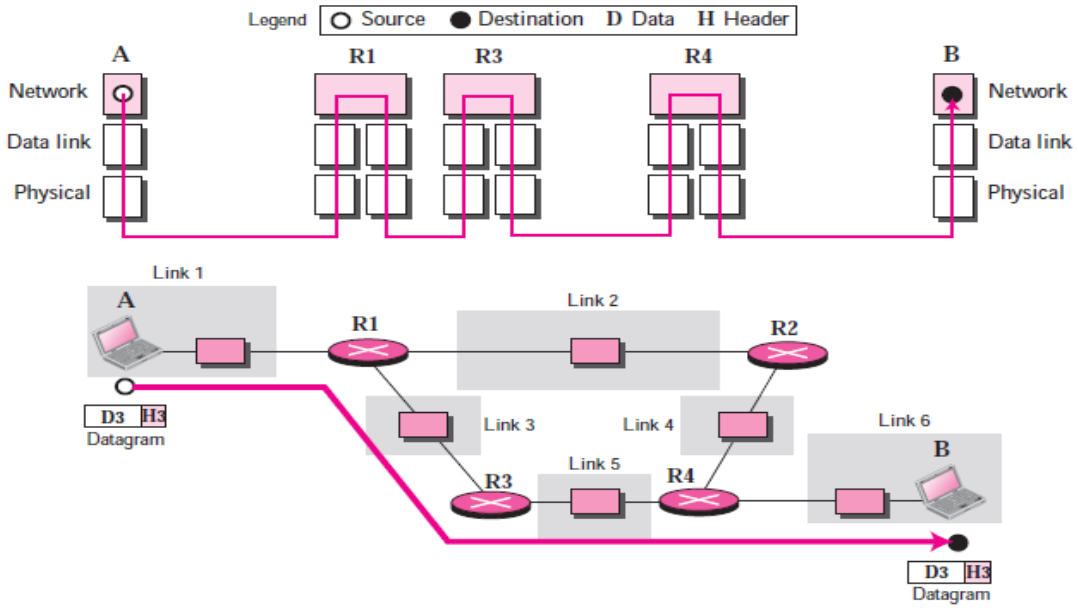


Figure – Communication at Datalink Layer

Network Layer:

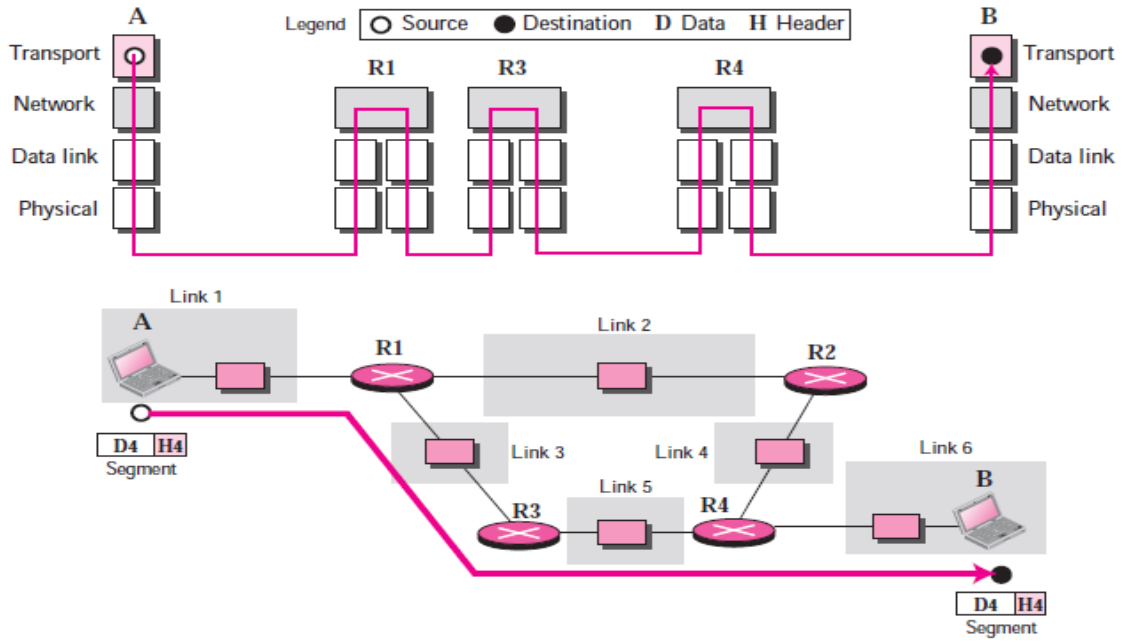
- TCP/IP supports the Internet Protocol (IP). The **Internet Protocol (IP)** is the transmission mechanism used by the TCP/IP protocols. IP transports data in packets called **datagrams**, each of which is transported separately.
- Datagrams can travel along different routes and can arrive out of sequence or be duplicated. IP does not keep track of the routes and has no facility for reordering datagrams once they arrive at their destination.
- Figure – shows the communication at the network layer.At the Network layer, TCP/IP supports the Internetworking Protocol (IP). There is a main difference between the communication at the network layer and the communication at data link or physical layers.
- Communication at the network layer is end to end while the communication at the other two layers are node to node. The datagram started at computer A is the one that reaches computer B. The network layers of the routers can inspect the source and destination of the packet for finding the best route, but they are not allowed to change the contents of the packet.
- Of course, the communication is logical, not physical. Although the network layer of computer A and B *think* that they are sending and receiving datagrams, the actual communication again is done at the physical level.



- IP in turn uses some supporting protocols such as ARP, RARP, ICMP and IGMP.
  -IP (Internetworking Protocol) is an unreliable and connectionless protocol. It gives a best effort delivery service.
  -ARP (Address Resolution protocol) is used to get physical address when logical address is known.
  -RARP (Reverse Address Resolution protocol) is used to get logical address when physical address is known.
  -ICMP (Internet Control Message Protocol) sends query and error reporting messages.
  -IGMP (Internet Group Message Protocol) is used to facilitate the simultaneous transmission of a message to a group of recipients.

Transport Layer:

- There is a main difference between the transport layer and the network layer. Although all nodes in a network need to have the network layer, only the two end computers need to have the transport layer. The network layer is responsible for sending individual datagrams from computer A to computer B; the transport layer is responsible for delivering the whole message, which is called a segment, a user datagram, or a packet, from A to B.
- A segment may consist of a few or tens of datagrams. The segments need to be broken into datagrams and each datagram has to be delivered to the network layer for transmission. Since the Internet defines a different route for each datagram, the datagrams may arrive out of order and may be lost.
- The transport layer at computer B needs to wait until all of these datagrams to arrive, assemble them and make a segment out of them. Figure – shows the communication at the transport layer.



- Traditionally, the transport layer was represented in the TCP/IP suite by two protocols: **User Datagram Protocol (UDP)** and **Transmission Control Protocol (TCP)**. A new protocol called **Stream Control Transmission Protocol (SCTP)** has been introduced in the last few years.

Application Layer –

- The application layer in TCP/IP is equivalent to the combined session, presentation, and application layers in the OSI model. The application layer allows a user to access the services of our private internet or the global Internet. Many protocols are defined at this layer to provide services such as electronic mail, file transfer, accessing the World Wide Web and so on.
- Figure – shows the communication at the application layer. Note that the communication at the application layer, like the one at the transport layer, is end to end. A message generated at computer A is sent to computer B without being changed during the transmission. The unit of communication at the application layer is a message.

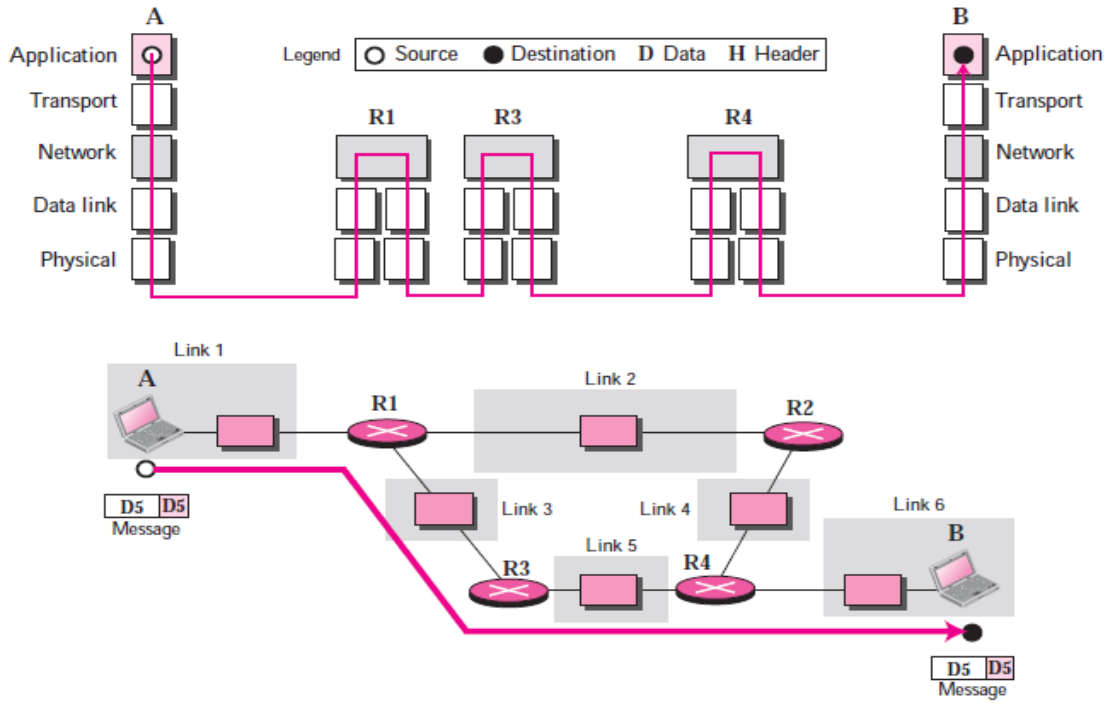
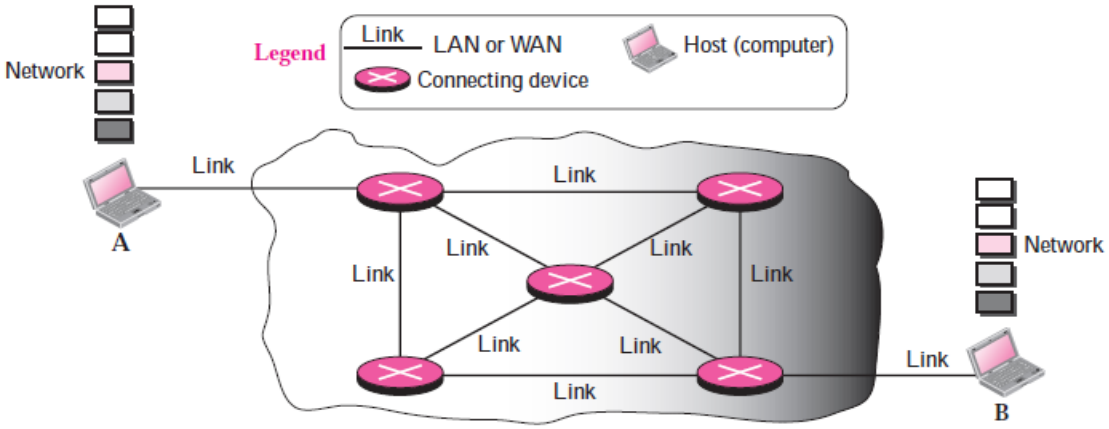


Figure – Communication in Application Layer

Network Layer –

- The Internet, however, is not one single network; it is made of many networks (or links) connected together through the connecting devices. In other words, the Internet is an internetwork, a combination of LANs and WANs.
- To better understand the role of the network layer (or the internetwork layer), we need to move from our conceptual level and think about all of these LANs and WANs that make the Internet. Since it is impossible to show all of these LANs and WANs, we show only an imaginary small internet with a few networks and a few connecting devices, as shown in Figure –



An internet as a combination of links and connecting devices

- In this model, a connecting device such as a router acts as a switch. When a packet arrives from one of its ports (interface), the packet is forwarded through another port to the next switch (or final destination). In other words, a process called **switching** occurs at the connecting device.

Circuit Switching

- One solution to the switching is referred to as **circuit switching**, in which a physical circuit (or channel) is established between the source and destination of the message before the delivery of the message. After the circuit is established, the entire message, is transformed from the source to the destination.
- The source can then inform the network that the transmission is complete, which allows the network to open all switches and use the links and connecting devices for another connection.
- The circuit switching was never implemented at the network layer; it is mostly used at the physical layer.
- In circuit switching, the whole message is sent from the source to the destination without being divided into packets.

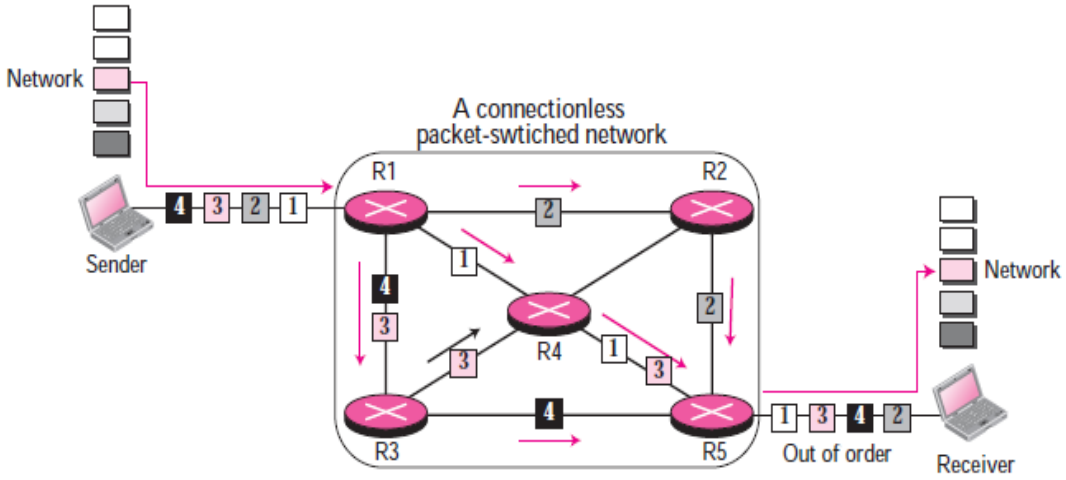
Packet Switching

- The second solution to switching is called **packet switching**. The network layer in the Internet today is a packet-switched network. In this type of network, a message from the upper layer is
- The source of the message sends the packets one by one; the destination of the message receives the packets one by one. The destination waits for all packets belonging to the same message to arrive before delivering the message to the upper layer.
- The connecting devices in a packet-switching network still need to decide how to route the packets to the final destination.
- In packet switching, the message is first divided into manageable packets at the source before being transmitted. The packets are assembled at the destination. A packet-switched network can use two different approaches to route the packets: **The Datagram Approach** and The **Virtual Circuit Approach.**

PACKET SWITCHING AT NETWORK LAYER –

Connectionless Service

When the Internet started, the network layer was designed to provide a **connectionless service**, in which the network layer protocol treats each packet independently, with each packet having no relationship to any other packet. The packets in a message may or may not travel the same path to their destination. When the Internet started, it was decided to make the network layer a connectionless service to make it simple. The network layer is only responsible for delivery of packets from the source to the destination. Figure – shows connectionless packet switch network.



- When the network layer provides a connectionless service, each packet travelling in the Internet is an independent entity; there is no relationship between packets belonging to the same message. The switches in this type of network are called *routers*.
- A packet belonging to a message may be followed by a packet belonging to the same message or a different message. A packet may be followed by a packet coming from the same or from a different source.
- Each packet is route based on the information contained in its header: source and destination address. The destination address defines where it should go; the source address defines where it comes from.
- The router in this case routes the packet based only on the destination address. The source address may be used to send an error message to the source if the packet is discarded. Figure – shows the forwarding process in a router in this case. We have used symbolic addresses such as A and B.

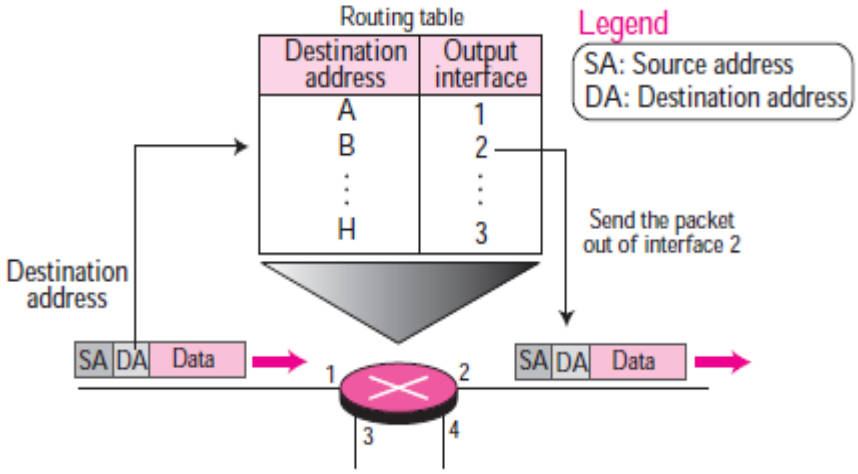


Figure - Forwarding process in a router when used in a connectionless network

In a connectionless packet-switched network, the forwarding decision is based on the destination address of the packet.

Delay In Connectionless Network –

If we ignore the fact that the packet may be lost and resent and also the fact that the destination may be needed to wait to receive all packets, we can model the delay as shown in Figure –

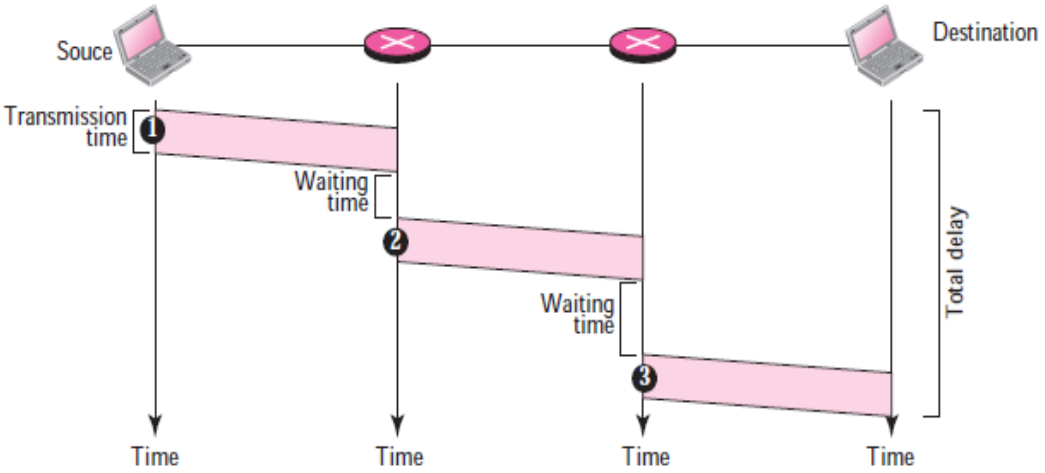
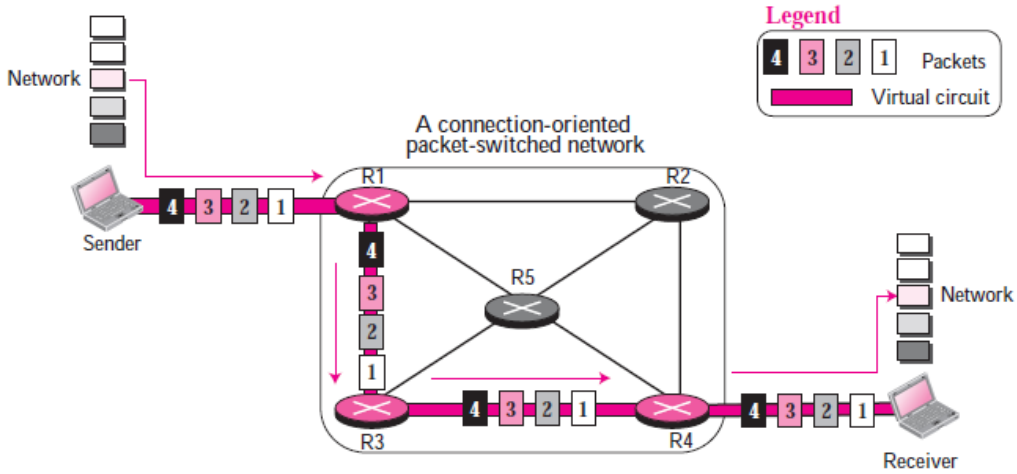


Figure - Delay in a connectionless network

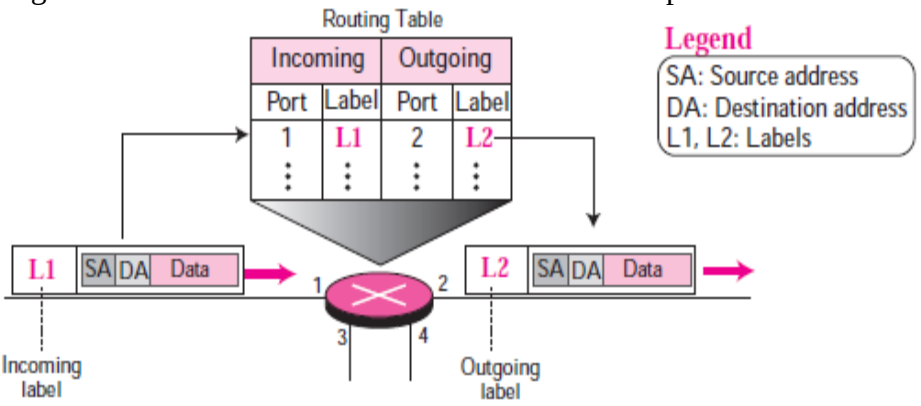
Connection-Oriented Service –

- In a **connection-oriented service,** there is a relation between all packets belonging to a message. Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams can follow the same path.
- In this type of service, not only must the packet contain the source and destination addresses, it must also contain a *flow label*, a *virtual circuit identifier* that defines the virtual path the packet should follow. Figure shows the concept of connection-oriented service.



- Each packet is forwarded based on the label in the packet. To follow the idea of connection-oriented design to be used in the Internet, we assume that the packet has a label when it reaches the router as shown in the figure –
- In this case, the forwarding decision is based on the value of the label, or virtual circuit identifier as it is sometimes called.

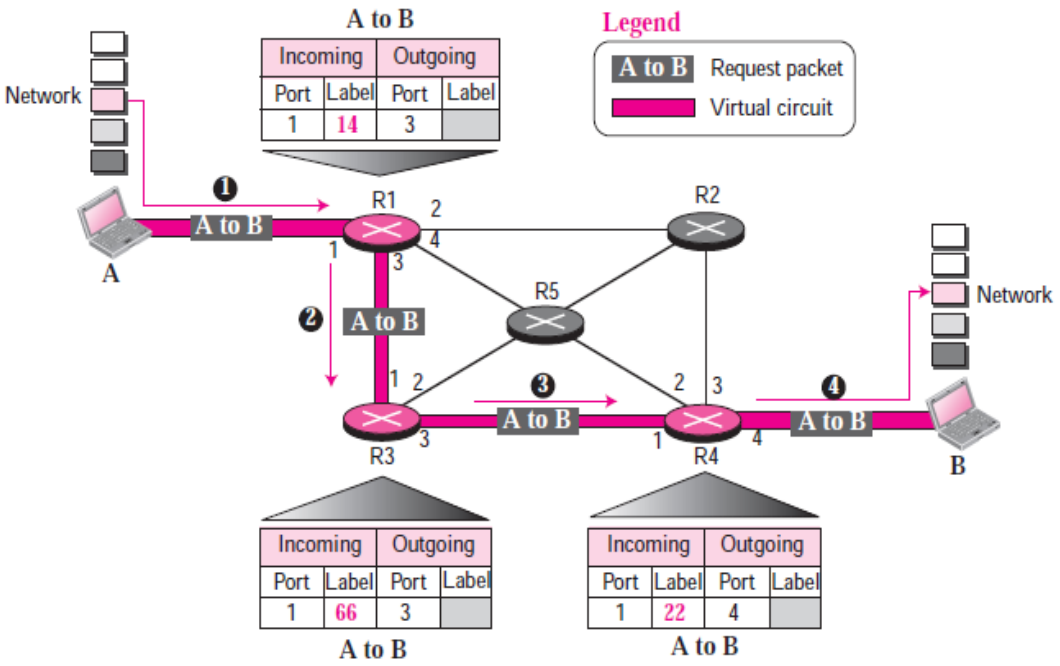
- To create a connection-oriented service, a three-phase process is used: *setup*, *data transfer*, and *teardown*. In the setup phase, the source and destination addresses of the sender and receiver is used to make table entries for the connection-oriented service. In the teardown phase, the source and destination inform the router to delete the corresponding entries. Data transfer occurs between these two phases.



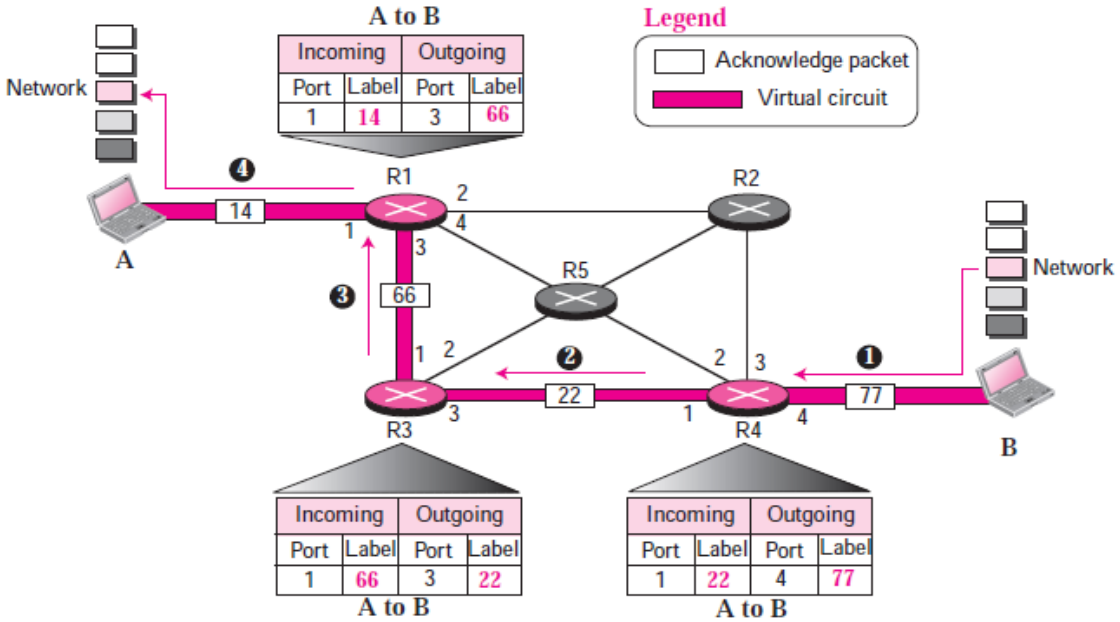
Setup Phase

In the **setup phase**, a router creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to destination B. Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.
Request packet A request packet is sent from the source to the destination. This auxiliary packet carries the source and destination addresses. Figure – shows the process.

- Source A sends a request packet to router R1.
- Router R1 receives the request packet. It knows that a packet going from A to B goes out through port How the router has obtained this information is a point covered in future chapters. For the moment, assume that it knows the output port. The router creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The router assigns the incoming port (1) and chooses an available incoming label (14) and the outgoing port (3). It does not yet know the outgoing label, which will be found during the acknowledgment step. The router then forwards the packet through port 3 to router R3.
- Router R3 receives the setup request packet. The same events happen here as at router R1; three columns of the table are completed: in this case, incoming port (1), incoming label (66), and outgoing port (2).
- Router R4 receives the setup request packet. Again, three columns are completed: incoming port (2), incoming label (22), and outgoing port (3).
- Destination B receives the setup packet, and if it is ready to receive packets from A, it assigns a label to the incoming packets that come from A, in this case 77. This label lets the destination know that the packets come from A, and not other sources.



Acknowledgment Packet A special packet, called the acknowledgment packet, completes the entries in the switching tables. Figure – shows the process.



1. The destination sends an acknowledgment to router R4. The acknowledgment carries the global source and destination addresses so the router knows which entry in the table is to be completed. The packet also carries label 77, chosen by the destination as the incoming label for packets from A. Router R4 uses this label to complete the outgoing label column for this entry. Note that 77 is the incoming label for destination B, but the outgoing label for router R4.

2. Router R4 sends an acknowledgment to router R3 that contains its incoming label in the table, chosen in the setup phase. Router R3 uses this as the outgoing label in the table.
3. Router R3 sends an acknowledgment to router R1 that contains its incoming label in the table, chosen in the setup phase. Router R1 uses this as the outgoing label in the table.
4. Finally router R1 sends an acknowledgment to source A that contains its incoming label in the table, chosen in the setup phase.
5. The source uses this as the outgoing label for the data packets to be sent to destination B.

Data Transfer Phase –

- The second phase is called the **data transfer phase**. After all routers have created their routing table for a specific virtual circuit, then the network-layer packets belonging to one message can be sent one after another.
- In Figure, we show the flow of one single packet, but the process is the same for 1, 2, or 100 packets. The source computer uses the label 14, which it has received from router R1 in the setup phase. Router R1 forwards the packet to router R3, but changes the label to 66. Router R3 forwards the packet to router R4, but changes the label to 22.
- Finally, router R4 delivers the packet to its final destination with the label 77. All the packets in the message follow the same sequence of labels to reach their destination. The packet arrives in order at the destination.

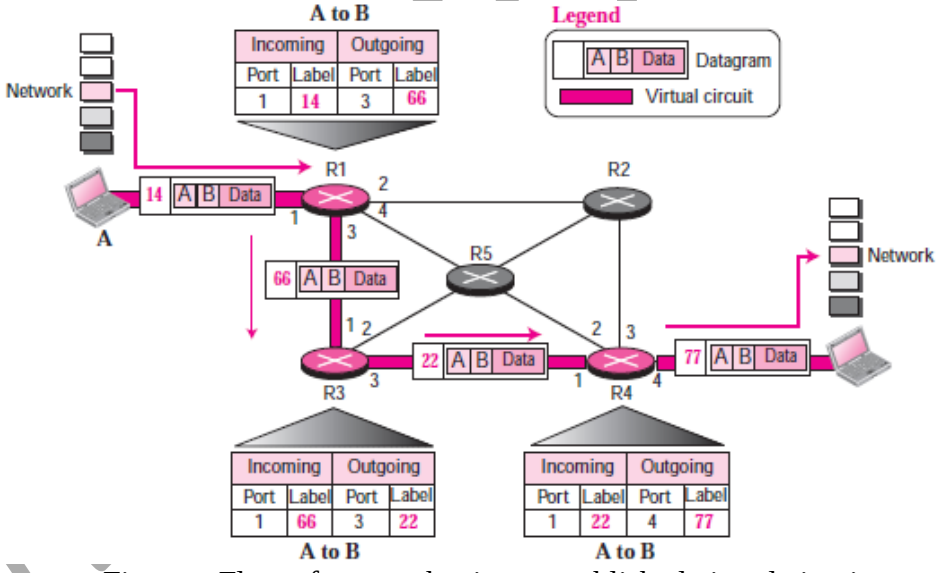


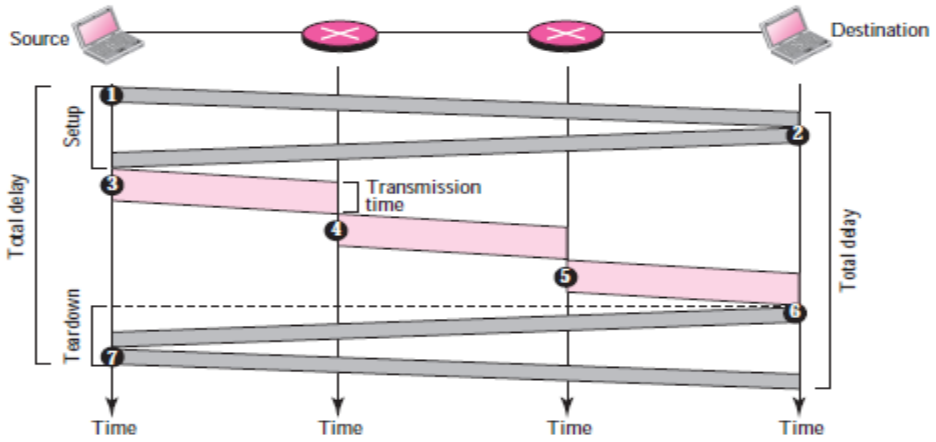
Figure – Flow of one packet in an established virtual circuit.

Teardown Phase –

In the **teardown phase,** source A, after sending all packets to B, sends a special packet called a teardown packet. Destination B responds with a confirmation packet. All routers delete the corresponding entry from their tables.

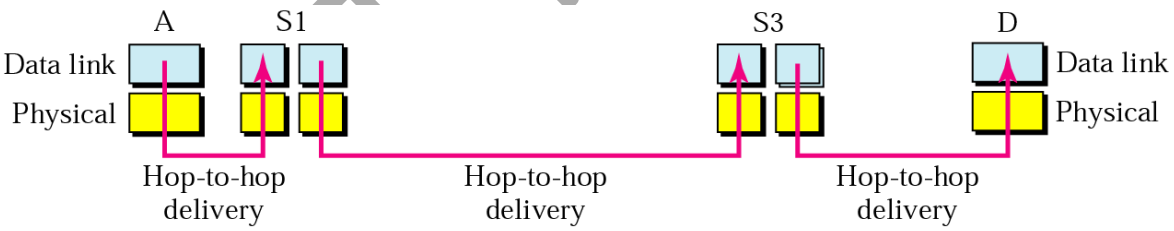
Delay In Connection-Oriented Network –

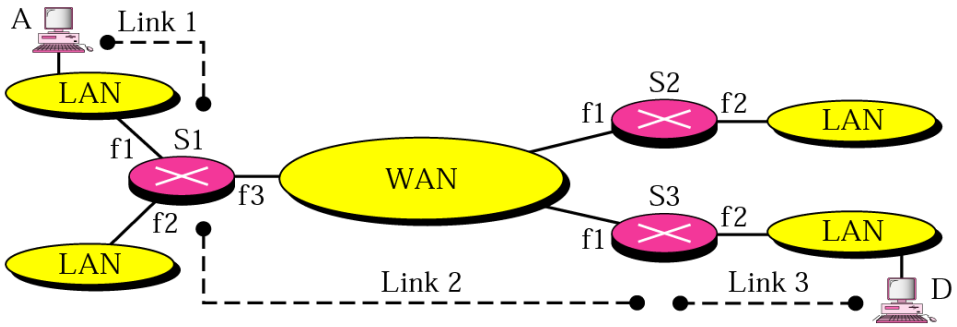
If we ignore the fact that the packet may be lost and resent, we can model the delay as shown in Figure –



Network Layer or Explain the need of Network Layer –
InterNetworking –

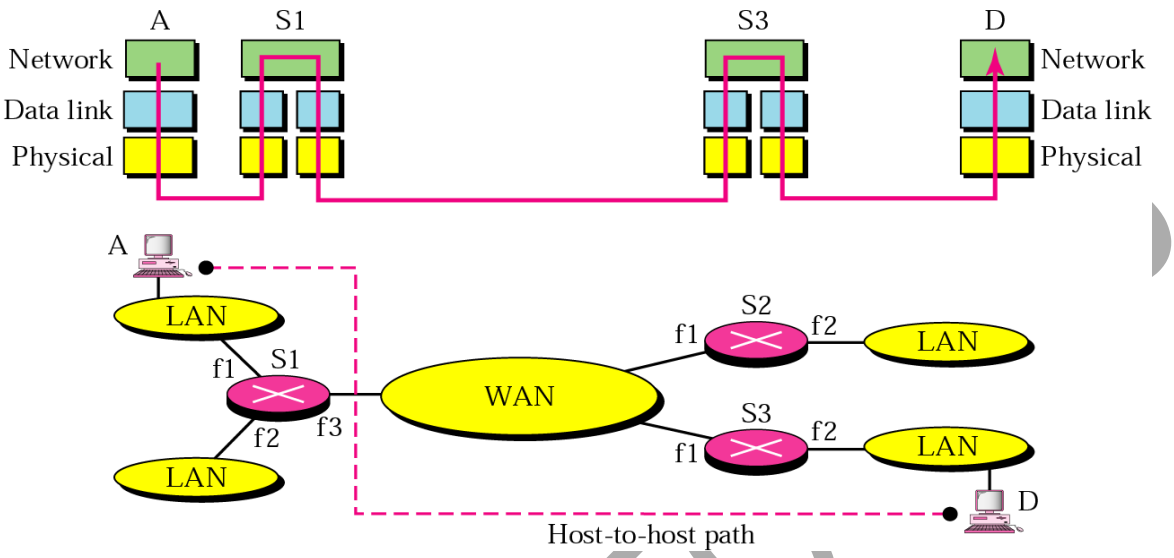
- The physical and data link layer of a network are responsible for data delivery on the network from one node to the next, as shown in figure –
- This internetwork is made of five networks: four LANs and one WAN. If host A needs to send a data packet to host D, the packet needs to go first from A to Rl (a switch or router), then from Rl to R3, and finally from R3 to host D.
- We say that the data packet passes through three links. In each link, two physical and two data link layers are involved. However, there is a big problem here. When data arrive at interface f1 of Rl, how does RI know that interface f3 is the outgoing interface? There is no provision in the data link (or physical) layer to help Rl make the right decision.
- The frame does not carry any routing information either. The frame contains the MAC address of A as the source and the MAC address of Rl as the destination. For a LAN or a WAN, delivery means carrying the frame through one link, and not beyond.





- To solve the problem of delivery through several links, the network layer was designed. It is also called Internetwork Layer.

- The network layer is responsible for host-to-host delivery and for routing the packets through the routers or switches. Figure – shows the same internetwork with a network layer.



- The general idea of the functionality of the network layer at a source, at a router, and at the destination. The network layer at the source is responsible for creating a packet from the data coming from another protocol (such as a transport layer protocol or a routing protocol).
- The header of the packet contains, among other information, the logical addresses of the source and destination. The network layer is responsible for checking its routing table to find the routing information (such as the outgoing interface of the packet or the physical address of the next node).
- If the packet is too large, the packet is fragmented. The network layer at the destination is responsible for address verification; it makes sure that the destination address on the packet is the same as the address of the host.
- If the packet is a fragment, the network layer waits until all fragments have arrived, and then reassembles them and delivers the reassembled packet to the transport layer.

IPv4 addresses –

- The identifier used in the IP layer of the TCP/IP protocol suite to identify each device connected to the Internet is called the Internet address or **IP address.** An IPv4 address is a 32-bit address that *uniquely* and *universally* defines the connection of a host or a router to the Internet; an IP address is the address of the interface.
- 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.

- However, if a device has two connections to the Internet, via two networks, it has two IPv4 addresses. The IPv4 addresses are *universal* in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space –

- A protocol like IPv4 that defines addresses has an **address space.** An address space is the total number of addresses used by the protocol. If a protocol uses *b* bits to define an address, the address space is 2*b* because each bit can have two different values (0 or 1).
- IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). Theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation –

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16).

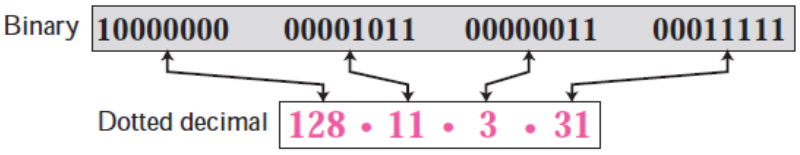
Binary Notation: Base 2 –

- In **binary notation,** an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces is usually inserted between each octet (8 bits). Each octet is often referred to as a byte. So it is common to hear an IPv4 address referred to as a 32-bit address, a 4-octet address, or a 4-byte address. The following is an example of an IPv4 address in binary notation:

```
01110101 10010101 00011101 11101010
```

Dotted-Decimal Notation: Base 256 –

- To make the IPv4 address more compact and easier to read, an IPv4 address is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as **dotted-decimal notation.** Figure 5.1 shows an IPv4 address in dotted decimal notation. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255.

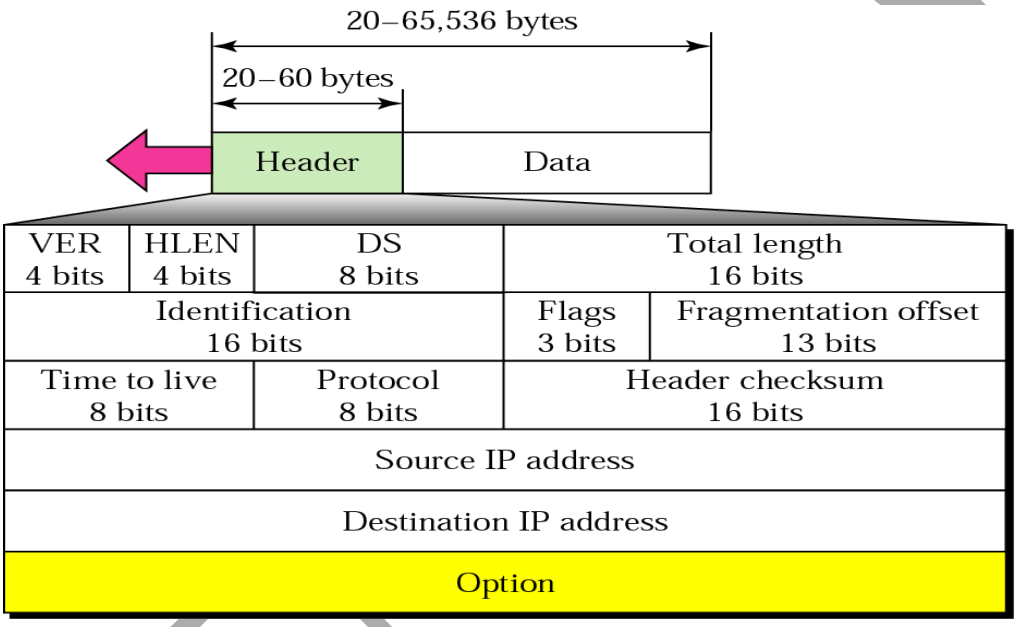


IPv4(Internet Protocol version 4)

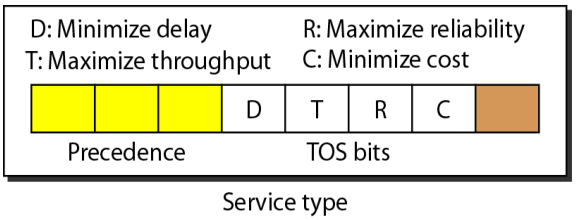
- Packets in the network (internet) layer are called **datagrams.** Figure 7.2 shows the IP datagram format. A datagram is a variable-length packet consisting of two parts: header and data. The header is 20 to 60 bytes in length and contains information essential to routing and delivery.

- The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/IP protocols. IPv4 is an unreliable and connectionless datagram protocol-a best-effort delivery service.
- The term *best-effort* means that IPv4 provides no error control or flow control. IPv4 assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.
- If reliability is important, IPv4 must be paired with a reliable protocol such as TCP.

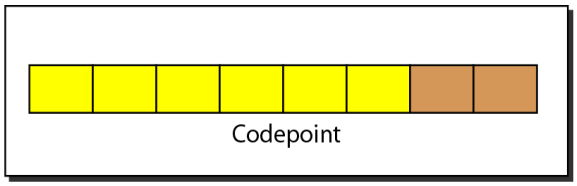
Datagram – Figure – shows the IPv4 datagram format.



- Version (VER) – This 4-bit field defines the version of the IPv4 protocol. Currently the version is 4. This field tells the IPv4 software running in the processing machine that the datagram has the format of version 4.
- Header length (HLEN) – This 4-bit field defines the total length of the datagram header in 4-byte words.This field is needed because the length of the header is variable (between 20 and 60 bytes).
- Services – IETF has changed the interpretation and name of this 8-bit field. This field, previously called service type, is now called differentiated services. We show both interpretations in Figure.



Service type



Differentiated services

1. Service Type

In this interpretation, the first 3 bits are called precedence bits. The next 4 bits are called type of service (TOS) bits, and the last bit is not used.
TOS bits is a 4-bit subfield with each bit having a special meaning. Although a bit can be either 0 or 1, one and only one of the bits can have the value of 1 in each datagram.
The bit patterns and their interpretations are given in Table – With only 1 bit set at a time, we can have five different types of services.

TOS Bits	*Description*
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

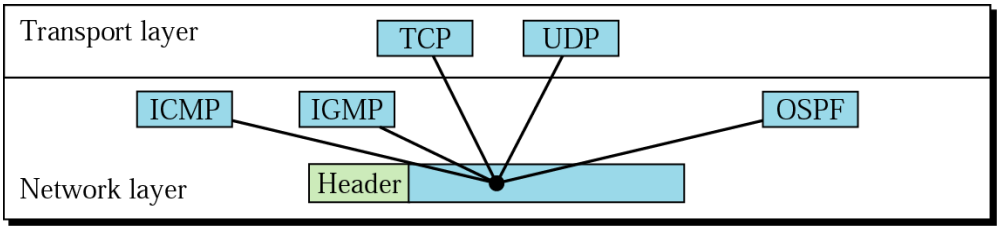
2. Differentiated Services
In this interpretation, the first 6 bits make up the codepoint subfield, and the last 2 bits are not used.

Table 7.1 *Values for codepoints*

Category	*Codepoint*	*Assigning Authority*
1	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental

- Total length. This is a In-bit field that defines the total length (header plus data) of the IPv4 datagram in bytes. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by 4.

  Length of data = total length - header length
- Identification – This field is used in fragmentation.
- Flags – This field is used in fragmentation.
- Fragmentation offset – This field is used in fragmentation.
- Time to live – A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero. However, for this scheme, all the machines must have synchronized clocks and must know how long it takes for a datagram to go from one machine to another.
- Protocol – This 8-bit field defines the higher-level protocol that uses the services of the IPv4 layer. An IPv4 datagram can encapsulate data from several higher-level protocols such as TCP, UDP, ICMP, and IGMP. This field specifies the final destination protocol to which the IPv4 datagram is delivered.



The value of this field for each higher-level protocol is shown in Table –

Value	*Protocol*
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

- Checksum – The checksum concept and its calculation are discussed later in thischapter.
- Source address – This 32-bit field defines the IPv4 address of the source. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.
- Destination address – This 32-bit field defines the IPv4 address of the destination. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

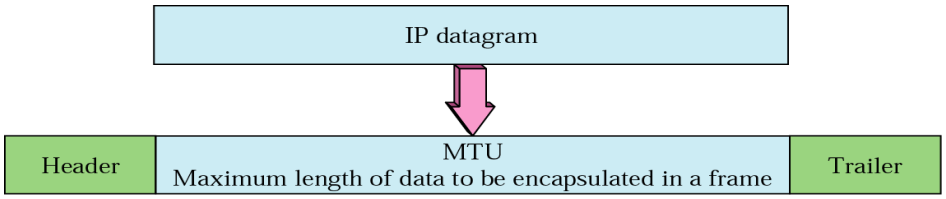
Explain the Fragmentation with MTU (Maximum Transfer Unit) –

Fragmentation

- A datagram can travel through different networks. Each router decapsulates the IPv4 datagram from the frame it receives, processes it, and then encapsulates it in another frame.
- The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel. For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU)

- Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field.
- In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network as shown in figure –



Frame

- The value of the MTU depends on the physical network protocol. Table – shows the values for some protocols.

Protocol	*MTU*
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

- To make the IPv4 protocol independent of the physical network, the designers decided to make the maximum length of the IPv4 datagram equal to 65,535 bytes.
- This makes transmission more efficient if we use a protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible to pass through these networks. This is called fragmentation.

Fields Related to Fragmentation –

The fields that are related to fragmentation and reassembly of an IPv4 datagram are the identification, flags, and fragmentation offset fields.

Identification –

- This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host.
- When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram.
- The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.

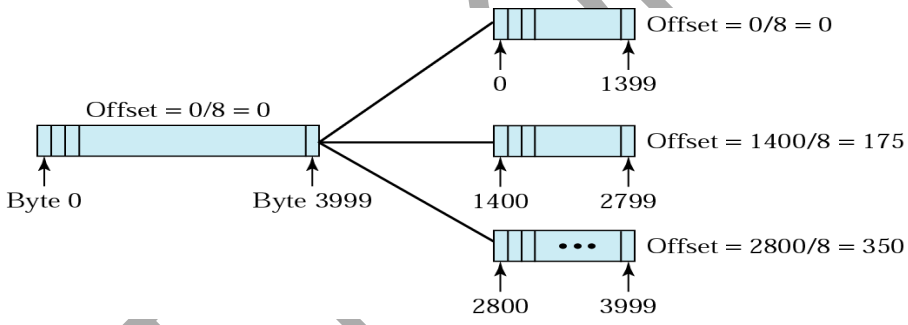
Flags – This is a 3-bit field.

1. The first bit is reserved.
2. The second bit is called the do not fragment bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host. If its value is 0, the datagram can be fragmented if necessary.

3. The third bit is called the more fragment bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment in Figure –

Fragmentation offset –

- This 13-bit field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes.
- Figure – shows a datagram with a data size of 4000 bytes fragmented into three fragments. The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 =O. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 =350.



Checksum –

The error detection method used by most TCP/IP protocols is called the **checksum.** The checksum protects against the corruption that may occur during the transmission of a packet. It is redundant information added to the packet.
The checksum is calculated at the sender and the value obtained is sent with the packet. The receiver repeats the same calculation on the whole packet including the checksum. If the result is satisfactory the packet is accepted; otherwise, it is rejected.

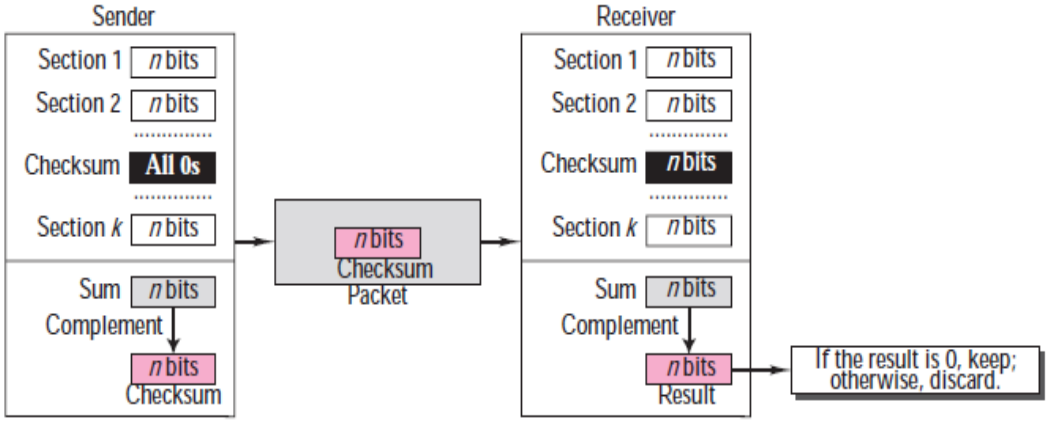
Checksum Calculation at the Sender –

- At the sender, the packet header is divided into *n*-bit sections (*n* is usually 16). These sections are added together using one's complement arithmetic, resulting in a sum that is also *n* bits long. The sum is then complemented (all 0s changed to 1s and all 1s to 0s) to produce the checksum.

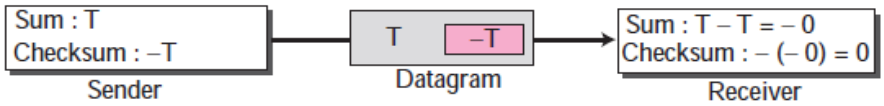
1. The packet is divided into *k* sections, each of *n* bits.
2. All sections are added together using one's complement arithmetic.
3. The final result is complemented to make the checksum.

Checksum Calculation at the Receiver –

- The receiver divides the received packet into *k* sections and adds all sections. It then complements the result. If the final result is 0, the packet is accepted; otherwise, it is rejected. Figure – shows graphically what happens at the sender and the receiver.
- We said when the receiver adds all of the sections and complements the result, it should get zero if there is no error in the data during transmission or processing.
- This is true because of the rules in one's complement arithmetic. Assume that we get a number called *T* when we add all the sections in the sender.
- When we complement the number in one's complement arithmetic, we get the negative of the number. This means that if the sum of all sections is *T*, the checksum is *–T*.



- When the receiver receives the packet, it adds all the sections. It adds *T* and *–T* which, in one's complement, is –0 (minus zero). When the result is complemented, –0 becomes 0. Thus if the final result is 0, the packet is accepted; otherwise, it is rejected see Figure –



IPv6 –

- The network layer protocol in the TCP/IP protocol suite is currently IPv4 (Internetworking Protocol, version 4). IPv4 provides the host-to-host communication between systems in the Internet.
- IPv4 has some deficiencies that make it unsuitable for the fast-growing Internet.
  1. Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
  2. The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
  3. The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided by IPv4.
- To overcome these deficiencies, IPv6 (Internetworking Protocol, version 6), also known as IPng (Internetworking Protocol, next generation).

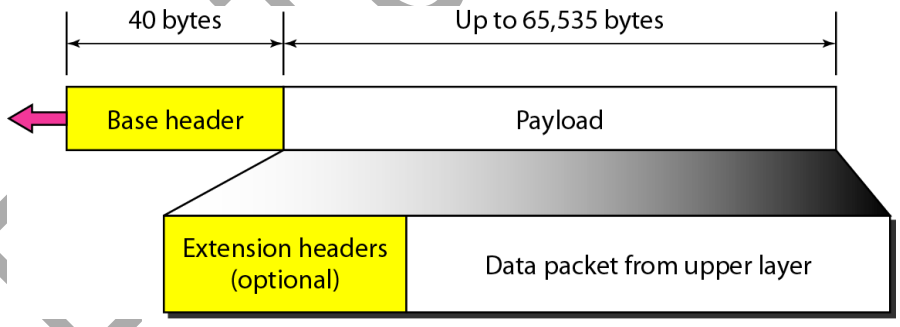
Advantages –

The next-generation IP, or IPv6, has some advantages over IPv4 that can be summarized as follows:

1. 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.
2. Better header format – IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
3. New options – IPv6 has new options to allow for additional functionalities.
4. Allowance for extension – IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
5. 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.
6. Support for more security – The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format –

The IPv6 packet is shown in Figure – The payload consists of two parts: optional extension headers and data from an upper layer. The base header occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.



Base Header –

Figure – shows the base header with its eight fields. These fields are as follows:

1. Version – This 4-bit field defines the version number of the IP. For IPv6, the value is 6.
2. Priority – The 4-bit priority field defines the priority of the packet with respect to traffic congestion.
3. Flow label – The flow label is a 3-byte (24-bit) field that is designed to provide special handling for a particular flow of data.
4. Payload length – The 2-byte payload length field defines the length of the IP datagram excluding the base header.

5. Next header – The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either one of the optional extension headers used by IP or the header of an encapsulated packet such as UDP or TCP. Each extension header also contains this field. Table – shows the values of next headers. Note that this field in version 4 is called the ***protocol.***

Code	*Next Header*
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

6. Hop limit – This 8-bit hop limit field serves the same purpose as the TTL field in IPv4.
7. Source address – The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.
8. Destination address – The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram.

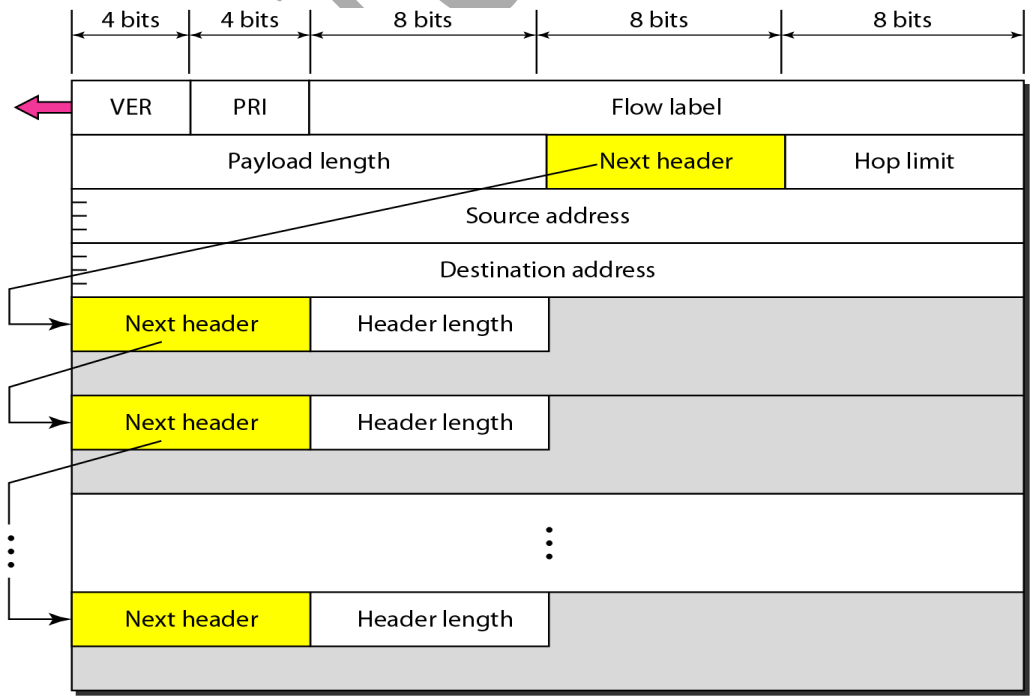


FIGURE – FORMAT OF AN IPV6 DATAGRAM

Unit-II

Address Mapping –

- An internet is made of a combination of physical networks connected by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host.
- The hosts and routers are recognized at the network level by their logical (IP) addresses. However, packets pass through physical networks to reach these hosts and routers. At the physical level, the hosts and routers are recognized by their physical addresses.
- A physical address is a local address.It must be unique locally, but is not necessarily unique universally. It is called a *physical* address because it is usually (but not always) implemented in hardware.
- An example of a physical address is the 48-bit MAC address in the Ethernet protocol, which is imprinted on the NIC installed in the host or router.
- The physical address and the logical address are two different identifiers. We need both because a physical network such as Ethernet can have two different protocols at the network layer such as IP and IPX (Novell) at the same time.

Static Mapping

Static mapping means creating a table that associates a logical address with a physical address. This table is stored in each machine on the network. Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table. This has some limitations because physical addresses may change in the following ways:
1. A machine could change its NIC, resulting in a new physical address.
2. In some LANs, such as LocalTalk, the physical address changes every time the computer is turned on.
3. A mobile computer can move from one physical network to another, resulting in a change in its physical address.

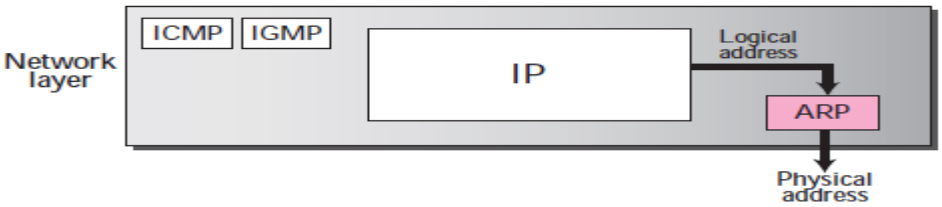
To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

Dynamic Mapping

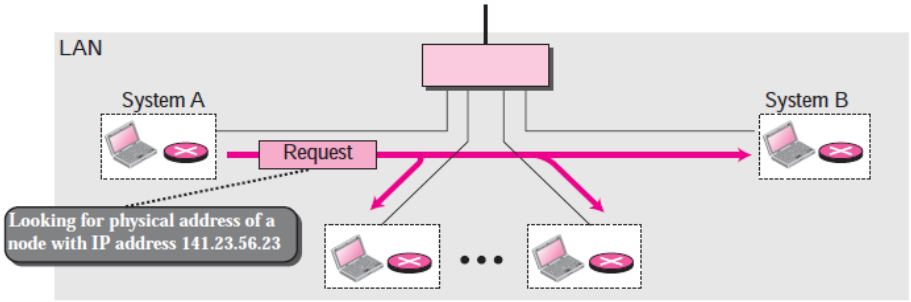
In **dynamic mapping**, each time a machine knows the logical address of another machine, it can use a protocol to find the physical address. Two protocols have been designed to perform dynamic mapping: **Address Resolution Protocol (ARP)** and **Reverse Address Resolution Protocol (RARP)**.

Mapping Logical to Physical Address: ARP(Address Resolution Protocol)
Explain ARP request is broadcast and ARP reply is unicast.

- 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.But the IP datagram must be encapsulated in a frame to be able to pass through the physical network.

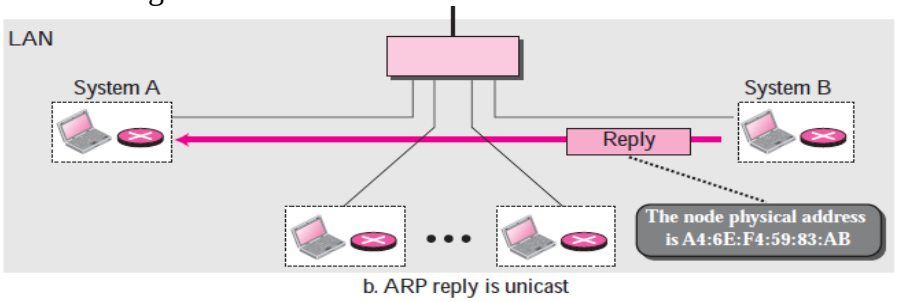


- This means that the sender needs the physical address of the receiver. The host or the router sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver.
- Because the sender does not know the physical address of the receiver, the query is broadcast over the network in figure –
- 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.



a. ARP request is multicast

- In Figure – a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address 141.23.56.23. System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient.
- It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IF address of 141.23.56.23. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure – b.

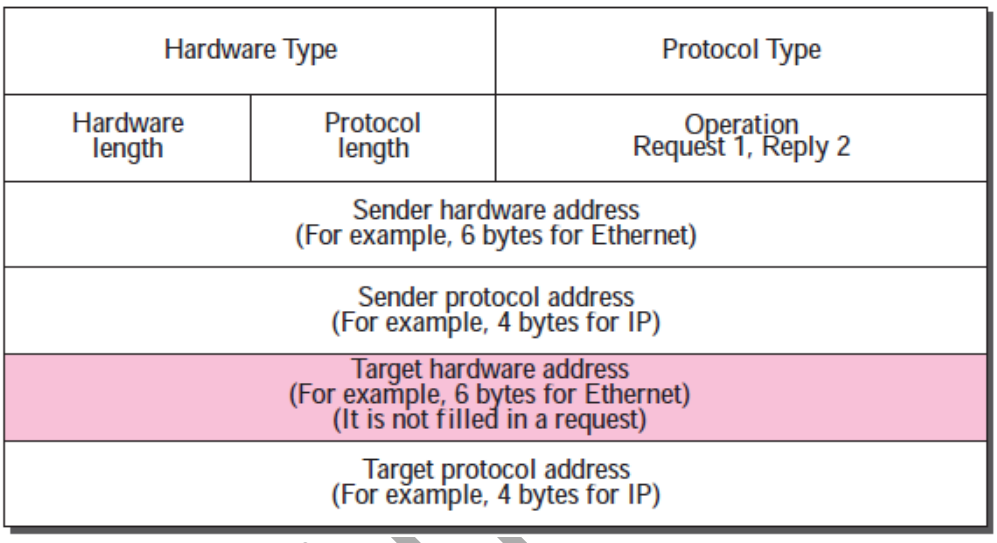


b. ARP reply is unicast

- System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination by using the physical address it received.

ARP Packet Format –

Figure 21.2 shows the format of an ARP packet.



The fields are as follows:

1. 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.
2. Protocol type. This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 0800_{16}. ARP can be used with any higher-level protocol.
3. 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.
4. 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.
5. Operation. This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1) and ARP reply (2).
6. 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.
7. 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.
8. Target hardware address. This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request

message, this field is alI 0s because the sender does not know the physical address of the target.

9. 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.

Explain the Encapsulation in ARP operation –

An ARP packet is encapsulated directly into a data link frame. For example, in Figure – an ARP packet is encapsulated in an Ethernet frame.

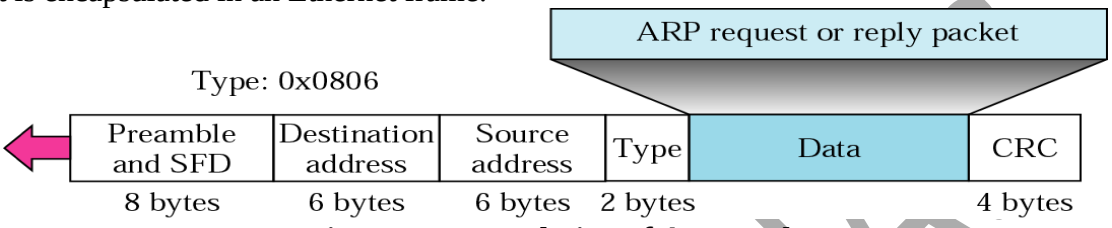


Figure – encapsulation of ARP packet

These are the steps involved in an ARP process:

1. The sender knows the IP address of the target.
2. IP asks ARP to create an ARP request message, filling in the sender physical address, the sender IP address, and the target IP address. The target physical address field is filled with 0s.
3. The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.
4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.
5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.
6. The sender receives the reply message. It now knows the physical address of the target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

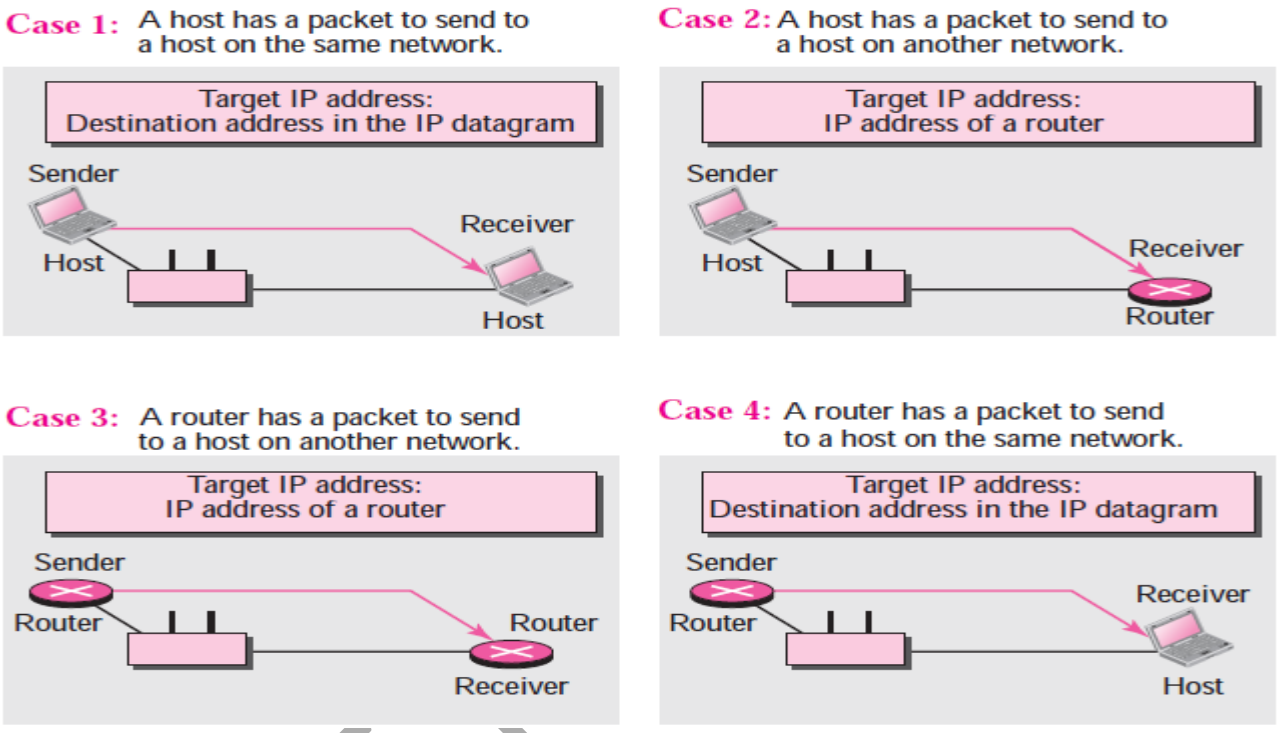
Four Different Cases –

The following are four different cases in which the services of ARP can be used in Figure –

Case 1: The sender is a host and wants to send a packet to another host on the same network. In this case, the logical address that must be mapped to a physical address is the destination IP address in the datagram header.

Case 2: The sender is a host and wants to send a packet to another host on another network. In this case, the host looks at its routing table and finds the IP address of the next hop (router) for this destination. If it does not have a routing table, it looks for the IP address of the default router. The IP address of the router becomes the logical address that must be mapped to a physical address.

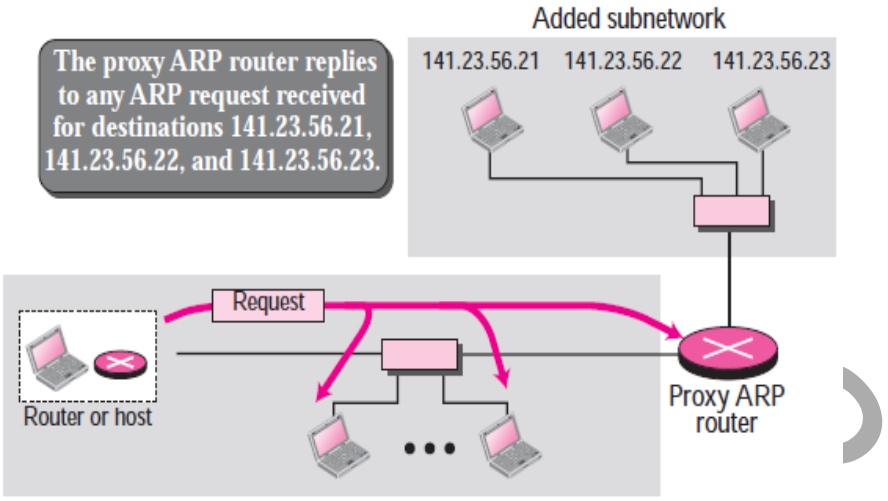
Case 3: The sender is a router that has received a datagram destined for a host on another network. It checks its routing table and finds the IP address of the next router. The IP address of the next router becomes the logical address that must be mapped to a physical address.
Case 4: The sender is a router that has received a datagram destined for a host in the same network. The destination IP address of the datagram becomes the logical address that must be mapped to a physical address.



Write a short notes on ProxyARP

How is the effect of sub – netting created in ARP

- A technique called *proxy* ARP is used to create a subnetting effect. A **proxy ARP** is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address.
- After the router receives the actual IP packet, it sends the packet to the appropriate host or router. Let us give an example. In Figure – the ARP installed on the right-hand host will answer only to an ARP request with a target IP address of 141.23.56.23.



- However, the administrator may need to create a subnet without changing the whole system to recognize subnetted addresses. One solution is to add a router running a proxy ARP.
- In this case, the router acts on behalf of all the hosts installed on the subnet. When it receives an ARP request with a target IP address that matches the address of one of its proteges (141.23.56.21, 141.23.56.22, or 141.23.56.23), it sends an ARP reply and announces its hardware address as the target hardware address.
- When the router receives the IP packet, it sends the packet to the appropriate host.

ATMARP –

- When IP packet are moving through an ATM WAN, a mechanism protocol is needed to find (map) the physical address of the exiting-point router in the ATM WAN given the IP address of the router.
- This is the same task performed by ARP on a LAN. However, there is a difference between a LAN and an ATM network.
- A LAN is a broadcast network (at the data link layer); ARP uses the broadcasting capability of a LAN to send (broadcast) an ARP request. An ATM network is not a broadcast network; another solution is needed to handle the task.

Packet Format –

The format of an **ATMARP** packet, which is similar to the ARP packet, is shown in Figure. The fields are as follows:

- **Hardware type (HTYPE).** The 16-bit HTYPE field defines the type of the physical network. Its value is 001316 for an ATM network.
- **Protocol type (PTYPE).** The 16-bit PTYPE field defines the type of the protocol. For IPv4 protocol the value is 080016. **Sender hardware length (SHLEN).** The 8-bit SHLEN field defines the length of the sender's physical address in bytes. For an ATM network the value is 20. Note that if the binding is done across an ATM network and two levels of hardware addressing are necessary, the neighboring 8-bit **reserved field** is used to define the length of the second address.

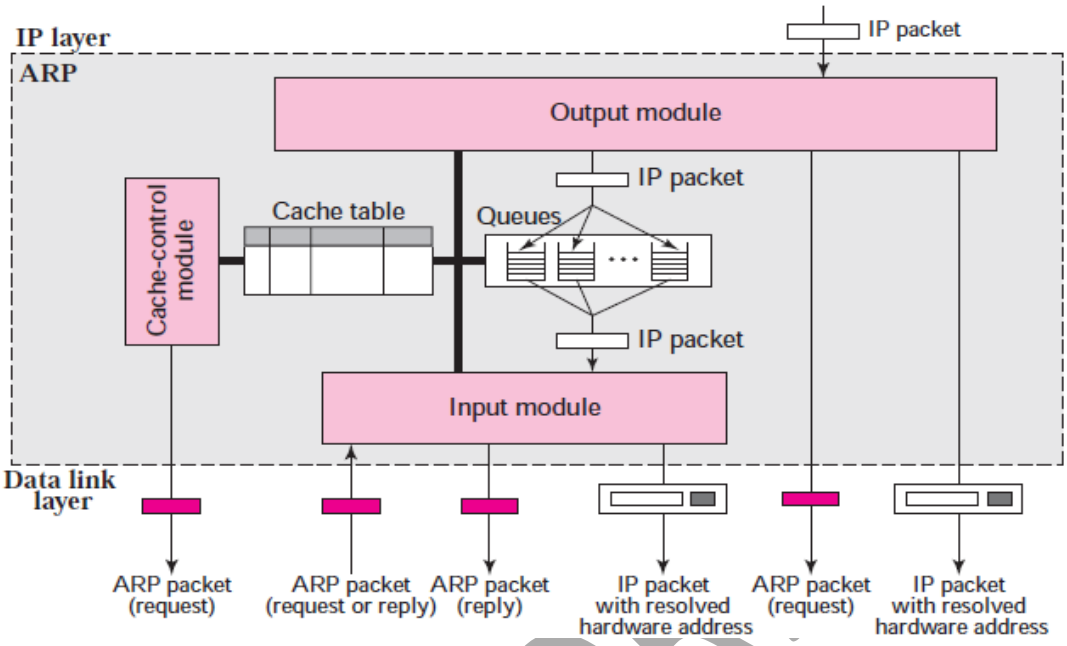
- **Operation (OPER).** The 16-bit OPER field defines the type of the packet. Five packet types are defined as shown in Table –

Message	*OPER value*
Request	1
Reply	2
Inverse Request	8
Inverse Reply	9
NACK	10

- **Sender protocol length (SPLEN).** The 8-bit SPLEN field defines the length of the address in bytes. For IPv4 the value is 4 bytes.
- **Target hardware length (TLEN).** The 8-bit TLEN field defines the length of the receiver's physical address in bytes. For an ATM network the value is 20. Note that if the binding is done across an ATM network and two levels of hardware addressing are necessary, the neighboring 8-bit reserved field is used to define the length of the second address.
- **Target protocol length (TPLEN).** The 8-bit TPLEN field defines the length of the address in bytes. For IPv4 the value is 4 bytes.
- **Sender hardware address (SHA).** The variable-length SHA field defines the physical address of the sender. For ATM networks defined by the ATM Forum, the length is 20 bytes.
- **Sender protocol address (SPA).** The variable-length SPA field defines the address of the sender. For IPv4 the length is 4 bytes.
- **Target hardware address (THA).** The variable-length THA field defines the physical address of the receiver. For ATM networks defined by the ATM Forum, the length is 20 bytes. This field is left empty for request messages and filled in for reply and NACK messages.
- **Target protocol address (TPA).** The variable-length TPA field defines the address of the receiver. For IPv4 the length is 4 bytes.

ARP PACKAGE –

- Figure – shows these components and their interactions. We can say that this ARP package involves five components: a cache table, queues, an output module, an input module, and a cache-control module.
- The package receives an IP datagram that needs to be encapsulated in a frame that needs the destination physical (hardware) address. If the ARP package finds this address, it delivers the IP packet and the physical address to the data link layer for transmission.



Cache Table –

When a host or router receives the corresponding physical address for an IP datagram, the address can be saved in the cache table. This address can be used for the datagrams destined for the same receiver within the next few minutes. However, as space in the cache table is very limited, mappings in the cache are not retained for an unlimited time. The cache table is implemented as an array of entries. In our package, each entry contains the following fields:

- **State.** This column shows the state of the entry. It can have one of three values: *FREE*, *PENDING*, or *RESOLVED*. The FREE state means that the time-to-live for this entry has expired. The space can be used for a new entry. The PENDING state means a request for this entry has been sent, but the reply has not yet been received. The RESOLVED state means that the entry is complete. The entry now has the physical (hardware) address of the destination. The packets waiting to be sent to this destination can use the information in this entry.
- **Hardware type.** This column is the same as the corresponding field in the ARP packet.
- **Protocol type.** This column is the same as the corresponding field in the ARP packet.
- **Hardware length.** This column is the same as the corresponding field in the ARP packet.
- **Protocol length.** This column is the same as the corresponding field in the ARP packet.
- **Interface number.** A router (or a multihomed host) can be connected to different networks, each with a different interface number. Each network can have different hardware and protocol types.
- **Queue number.** ARP uses numbered queues to enqueue the packets waiting for address resolution. Packets for the same destination are usually enqueued in the same queue.
- **Attempts.** This column shows the number of times an ARP request is sent out for this entry.
- **Time-out.** This column shows the lifetime of an entry in seconds.

- **Hardware address.** This column shows the destination hardware address. It remains empty until resolved by an ARP reply.
- **Protocol address.** This column shows the destination IP address.

Queues –

- Our ARP package maintains a set of queues, one for each destination, to hold the IP packets while ARP tries to resolve the hardware address. The output module sends unresolved packets into the corresponding queue.

Output Module –

- The output module waits for an IP packet from the IP software. The output module checks the cache table to find an entry corresponding to the destination IP address of this packet. The destination IP address of the IP packet must match the protocol address of the entry.
- If the entry is found and the state of the entry is RESOLVED, the packet along with the destination hardware address is passed to the data link layer for transmission.
- If the entry is found and the state of the entry is PENDING, the packet waits until the destination hardware address is found. Because the state is PENDING, there is a queue already created for this destination. The module sends the packet to this queue.
- If no entry is found, the module creates a queue and enqueues the packet. A new entry with the state of PENDING is created for this destination and the value of the ATTEMPTS field is set to 1. An ARP request packet is then broadcast.

Input Module –

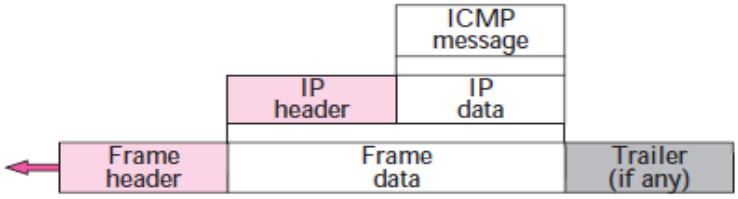
- The input module waits until an ARP packet (request or reply) arrives. The input module checks the cache table to find an entry corresponding to this ARP packet. The target protocol address should match the protocol address of the entry.
- If the entry is found and the state of the entry is PENDING, the module updates the entry by copying the target hardware address in the packet to the hardware address field of the entry and changing the state to RESOLVED. The module also sets the value of the TIME-OUT for this entry.
- If the entry is found and the state is RESOLVED, the module still updates the entry. This is because the target hardware address could have been changed. The value of the TIME-OUT field is also reset.
- If the entry is not found, the module creates a new entry and adds it to the table. The protocol requires that any information received is added to the table for future use. The state is set to RESOLVED and TIME-OUT is set.
- Now the module checks to see if the arrived ARP packet is a request. If it is, the module immediately creates an ARP reply message and sends it to the sender. The ARP reply packet is created by changing the value of the operation field from request to reply and filling in the target hardware address.

Cache-Control Module –

- The cache-control module is responsible for maintaining the cache table. It periodically (for example, every 5 s) checks the cache table, entry by entry. If the state of the entry is FREE, it continues to the next entry.
- If the state is PENDING, the module increments the value of the attempts field by 1. It then checks the value of the attempts field. If this value is greater than the maximum number of attempts allowed, the state is changed to FREE and the corresponding queue is destroyed.
- If the state of the entry is RESOLVED, the module decrements the value of the time-out field by the amount of time elapsed since the last check. If this value is less than or equal to zero, the state is changed to FREE and the queue is destroyed.

ICMP (Internet Control Message Protocol) –

- ICMP itself is a network layer protocol. However, its messages are not passed directly to the data link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer as shown in figure –



- The IP provides unreliable and connectionless datagram delivery. It was designed this way to make efficient use of network resources.
- The IP protocol is a best-effort delivery service that delivers a datagram from its original source to its final destination. However, it has two deficiencies: lack of error control and lack of assistance mechanisms.
- To solve this two deficiencies the Internet Control Message Protocol is designed.

Types of Messages –

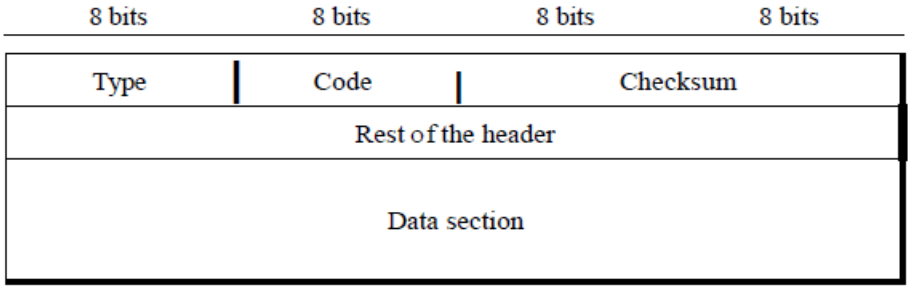
ICMP messages are divided into two broad categories: error-reporting messages and query messages.

- The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet.
- The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host.
- For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network, and routers can help a node redirect its messages.

Message Format –

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all.

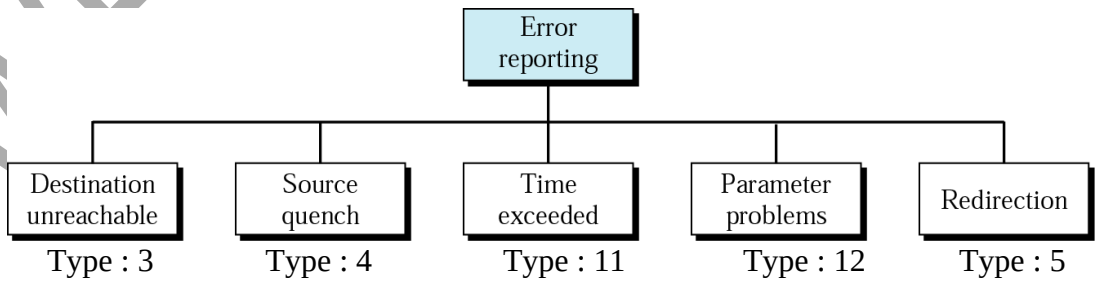
- ICMP type – defines the type of the message.
- Code – It specifies the reason for the particular message type.
- Checksum – It is used to detecting error.
- The rest of the header is specific for each message type.
- The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of the query.



Error Reporting –

One of the main responsibilities of ICMP is to report errors. However, ICMP does not correct errors-it simply reports them. Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses. ICMP uses the source IP address to send the error message to the source (originator) of the datagram.

The figure shows different types of error reporting –



Destination Unreachable – When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a destination-unreachable message back to the source host that initiated the datagram.

Source Quench – When a router or host discards a datagram due to congestion, it sends a source-quench message to the sender of the datagram. This message has two purposes. First, it informs the source that the datagram has been discarded. Second, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process.

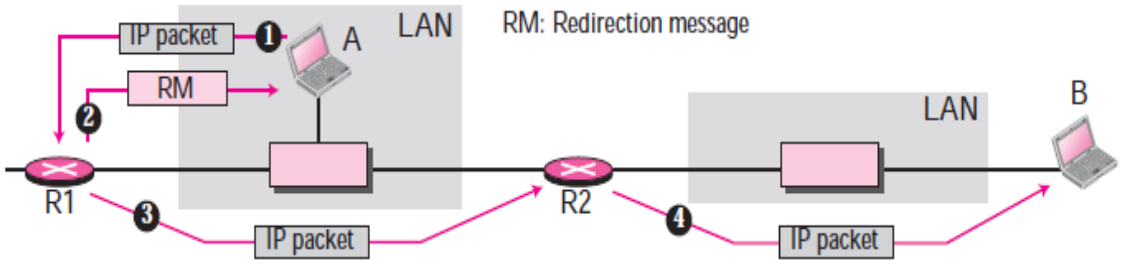
Time Exceeded –

- The time-exceeded message is generated in two cases: first, routers use routing tables to find the next hop (next router) that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loop or a cycle, going from one router to the next or visiting a series of routers endlessly.
- Each datagram contains a field called *time to live* that controls this situation. When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source.
- Second, a time-exceeded message is also generated when not all fragments that make up a message arrive at the destination host within a certain time limit.

Parameter Problem – If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.

Redirection –

- Updating the routing tables of hosts dynamically produces unacceptable traffic. The hosts usually use static routing. When a host comes up, its routing table has a limited number of entries. It usually knows the IP address of only one router, the default router.
- For this reason, the host may send a datagram, which is destined for another network, to the wrong router. In this case, the router that receives the datagram will forward the datagram to the correct router. However, to update the routing table of the host, it sends a redirection message to the host.
- This concept of redirection is shown in Figure – Host A wants to send a datagram to host B.

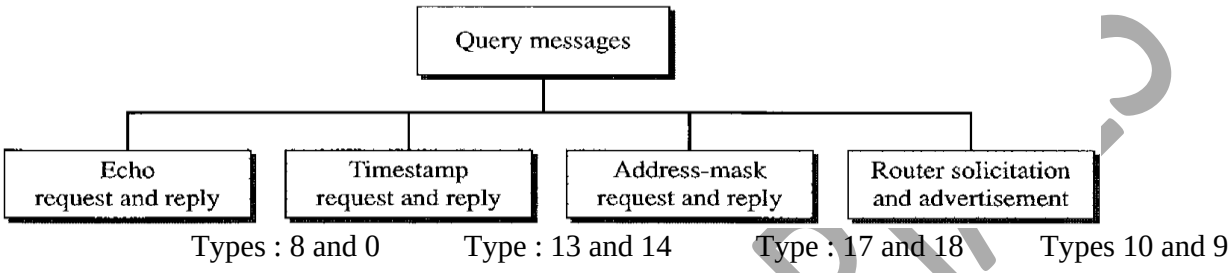


- Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead. Router R1, after consulting its table, finds that the packet should have gone to R2.

- It sends the packet to R2 and, at the same time, sends a redirection message to host A. Host A's routing table can now be updated.

Query –

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages, as shown in Figure –
In this type of ICMP message, a node sends a message that is answered in a specific format by the destination node.



Echo Request and Reply –

- The echo-request and echo-reply messages are designed for diagnostic purposes. The combination of echo-request and echo-reply messages determines whether two systems (hosts or routers) can communicate with each other.
- The echo-request and echo-reply messages can be used to determine if there is communication at the IP level. Because ICMP messages are encapsulated in IP datagrams, the receipt of an echo-reply message by the machine that sent the echo request is proof that the IP protocols in the sender and receiver are communicating with each other using the IP datagram. Also, it is proof that the intermediate routers are receiving, processing, and forwarding IP datagrams.

Timestamp Request and Reply –

Two machines (hosts or routers) can use the timestamp request and timestamp reply messages to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines.

It can also be used to synchronize the clocks in two machines.

Address-Mask Request and Reply –

- A host may know its IP address, but it may not know the corresponding mask. For example, a host may know its IP address as 159.31.17.24, but it may not know that the corresponding mask is /24. To obtain its mask, a host sends an address-mask-request message to a router on the LAN.
- If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host.

- This can be applied to its full IP address to get its subnet address.

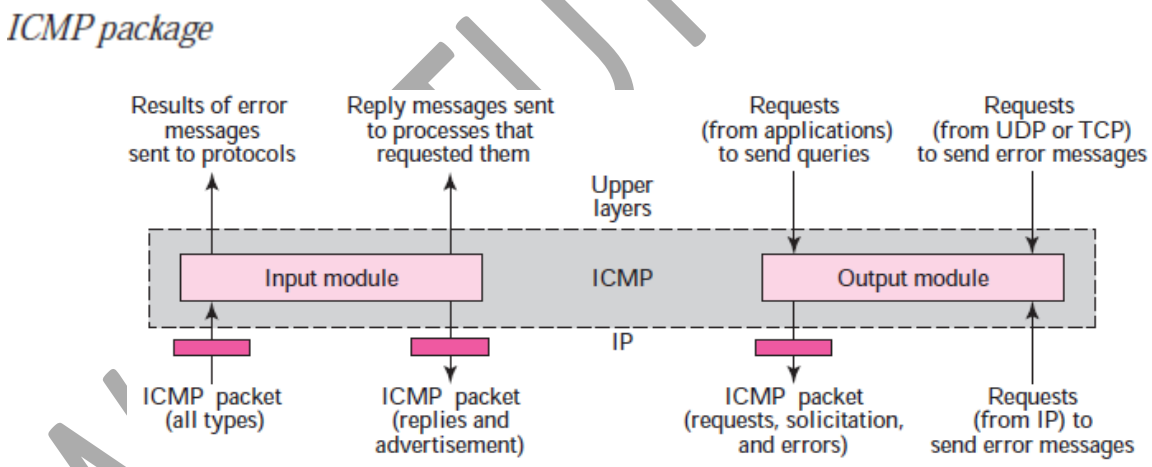
Router Solicitation and Advertisement –

- A host that wants to send data to a host on another network needs to know the address of routers connected to its own network. Also, the host must know if the routers are alive and functioning. The router-solicitation and router-advertisement messages can help in this situation.
- A host can broadcast a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited.
- Note that when a router sends out an advertisement, it announces not only its own presence but also the presence of all routers on the network of which it is aware.

ICMP PACKAGE –

To give an idea of how ICMP can handle the sending and receiving of ICMP messages, we present our version of an ICMP package made of two modules: an input module and an output module.

Figure – shows these two modules.



Input Module –

- The input module handles all received ICMP messages. It is invoked when an ICMP packet is delivered to it from the IP layer.
- If the received packet is a request, the module creates a reply and sends it out. If the received packet is a redirection message, the module uses the information to update the routing table. If the received packet is an error message, the module informs the protocol about the situation that caused the error.

Output Module –

- The output module is responsible for creating request, solicitation, or error messages requested by a higher level or the IP protocol. The module receives a demand from IP, UDP, or TCP to send one of the ICMP error messages.
- If the demand is from IP, the output module must first check that the request is allowed. Remember, an ICMP message cannot be created for four situations: an IP packet carrying an ICMP error message, a fragmented IP packet, a multicast IP packet, or an IP packet having IP address 0.0.0.0 or 127.X.Y. Z. The output module may also receive a demand from an application program to send one of the ICMP request messages.

MobileIP –

- Mobile communication has received a lot of attention in the last decade. The interest in mobile communication on the Internet means that the IP protocol, originally designed for stationary devices, must be enhanced to allow the use of mobile computers, computers that move from one network to another.

ADDRESSING

- The main problem that must be solved in providing mobile communication using the IP protocol is addressing.

Stationary Hosts

- The original IP addressing was based on the assumption that a host is stationary, attached to one specific network. A router uses an IP address to route an IP datagram.
- This implies that a host in the Internet does not have an address that it can carry with itself from one place to another. The address is valid only when the host is attached to the network. If the network changes, the address is no longer valid.
- The IP addresses are designed to work with stationary hosts because part of the address defines the network to which the host is attached.

Mobile Hosts:

When a host moves from one network to another, the IP addressing structure needs to be modified. Several solutions have been proposed.

Changing the Address

One simple solution is to let the **mobile host** change its address as it goes to the new network. The host can use DHCP to obtain a new address to associate it with the new network. This approach has several drawbacks.

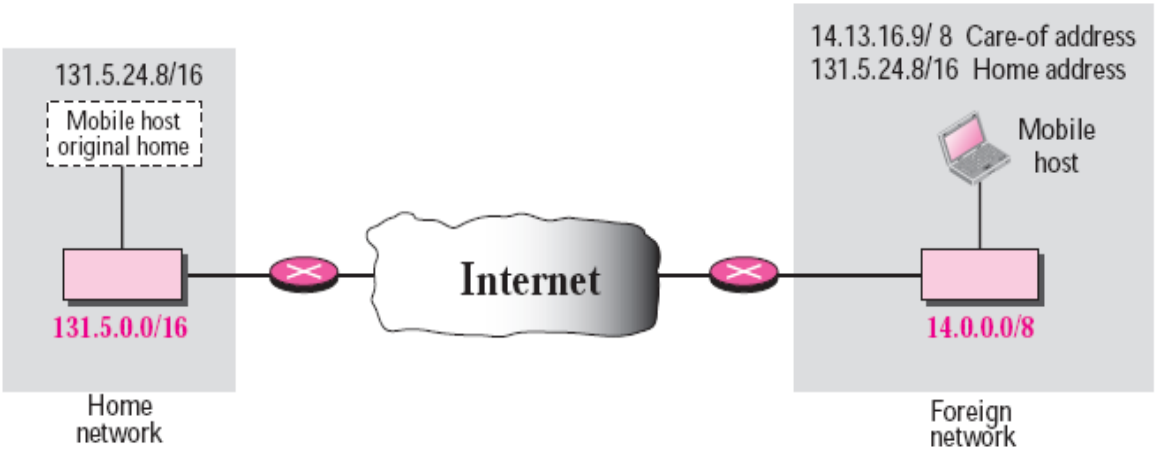
- First, the configuration files would need to be changed.
- Second, each time the computer moves from one network to another, it must be rebooted.

- Third, the DNS tables need to be revised so that every other host in the Internet is aware of the change.
- Fourth, if the host roams from one network to another during a transmission, the data exchange will be interrupted. This is because the ports and IP addresses of the client and the server must remain constant for the duration of the connection.

Two Addresses

The approach that is more feasible is the use of two addresses. The host has its original address, called the **home address,** and a temporary address, called the **care-of address.**

The home address is permanent; it associates the host to its **home network,** the network that is the permanent home of the host. The care-of address is temporary. When a host moves from one network to another, the care-of address Changes; it is associated with the **foreign network,** the network to which the host moves.



Mobile IP has two addresses for a mobile host: one home address and one care-of address. The home address is permanent; the care-of address changes as the mobile host moves from one network to another.

When a mobile host visits a foreign network, it receives its care-of address during the agent discovery and registration phase

AGENTS

To make the change of address transparent to the rest of the Internet requires a **home agent** and a **foreign agent.**
Figure shows the position of a home agent relative to the home network and a foreign agent relative to the foreign network.

Home agent and foreign agent

Home Agent

The home agent is usually a router attached to the home network of the mobile host. The home agent acts on behalf of the mobile host when a remote host sends a packet to the mobile host. The home agent receives the packet and sends it to the foreign agent.

Foreign Agent

The foreign agent is usually a router attached to the foreign network. The foreign agent receives and delivers packets sent by the home agent to the mobile host. The mobile host can also act as a foreign agent. In other words, the mobile host and the foreign agent can be the same. However, to do this, a mobile host must be able to receive a care-of address by itself, which can be done through the use of DHCP.

In addition, the mobile host needs the necessary software to allow it to communicate with the home agent and to have two addresses: its home address and its care-of address. This dual addressing must be transparent to the application programs. When the mobile host acts as a foreign agent, the care-of address is called a **colocated care-of address.**

The advantage of using a colocated care-of address is that the mobile host can move to any network without worrying about the availability of a foreign agent. The disadvantage is that the mobile host needs extra software to act as its own foreign agent.

THREE PHASES

To communicate with a remote host, a mobile host goes through three phases:

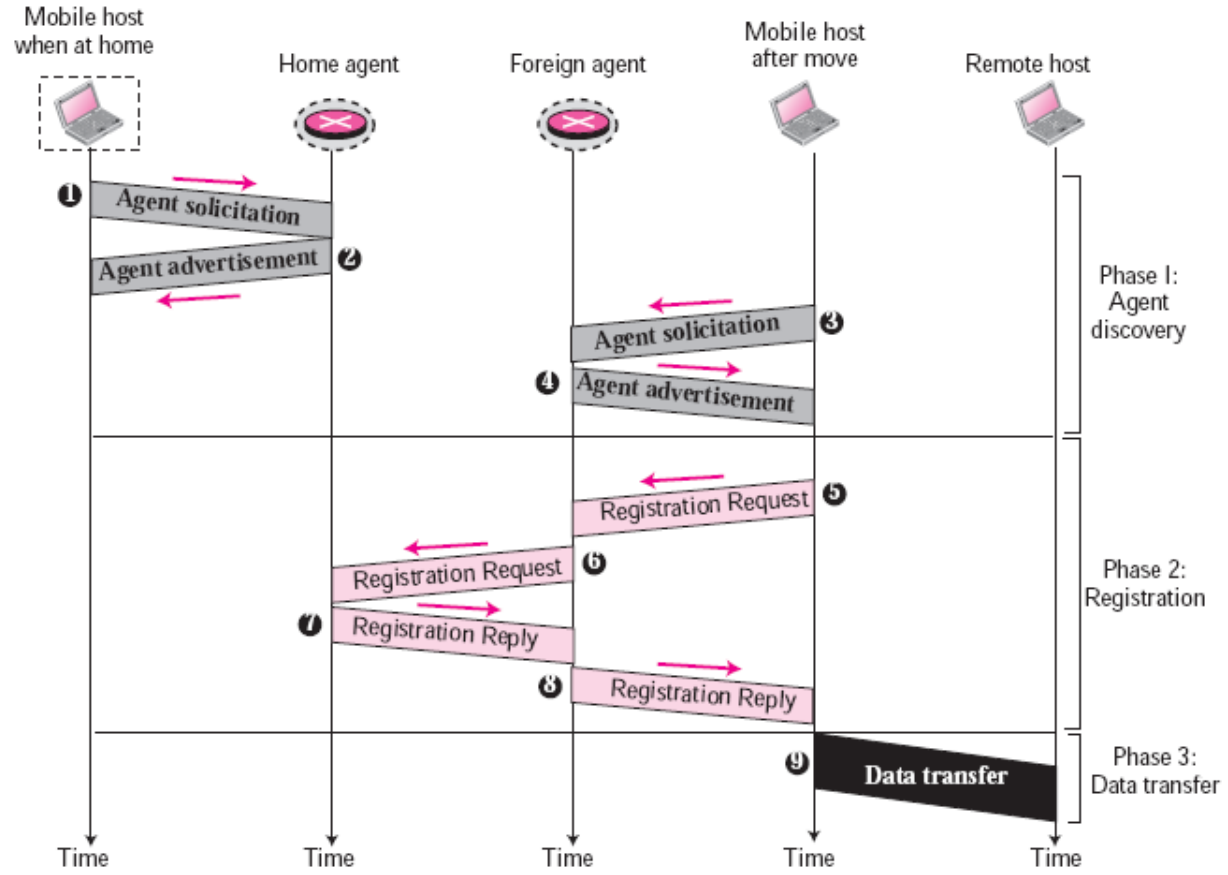
1. agent discovery
2. registration
3. data transfer

1. The first phase, **agent discovery**, involves the mobile host, the foreign agent, and the home agent.
2. The second phase, **registration**, also involves the mobile host and the two agents.
3. Finally, in the **third phase** data transfer , the remote host is also involved.

Agent Discovery

The first phase in mobile communication, **agent discovery,** consists of two subphases. A mobile host must discover (learn the address of) a home agent before it leaves its home network. A mobile host must also discover a foreign agent after it has moved to a foreign network. This discovery consists of learning the care-of address as well as the foreign agent's address. The discovery involves two types of messages: advertisement and solicitation.

Remote host and mobile host communication

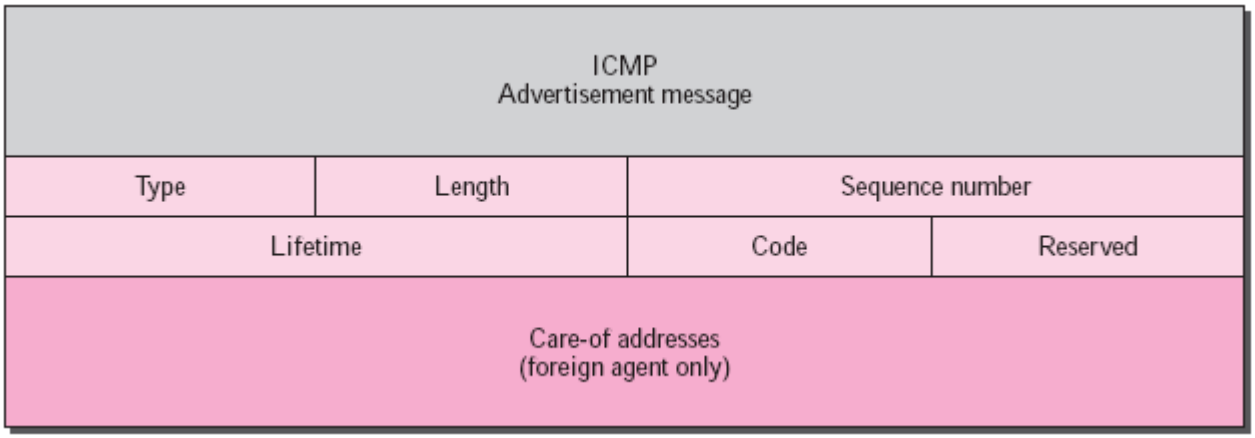


Agent Advertisement

When a router advertises its presence on a network using an ICMP router advertisement, it can append an **agent advertisement** to the packet if it acts as an agent.

Mobile IP does not use a new packet type for agent advertisement; it uses the router advertisement packet of ICMP, and appends an agent advertisement message.

Agent advertisement



The field descriptions are as follows:

- **Type.** The 8-bit type field is set to 16.
- **Length.** The 8-bit length field defines the total length of the extension message (not the length of the ICMP advertisement message).
- **Sequence number.** The 16-bit sequence number field holds the message number. The recipient can use the sequence number to determine if a message is lost.
- **Lifetime.** The lifetime field defines the number of seconds that the agent will accept requests. If the value is a string of 1s, the lifetime is infinite.
- **Code.** The code field is an 8-bit flag in which each bit is set (1) or unset (0).
- **Care-of Addresses.** This field contains a list of addresses available for use as careofaddresses. The mobile host can choose one of these addresses. The selection of this care-of address is announced in the registration request. Note that this field is used only by a foreign agent.

Agent Solicitation

When a mobile host has moved to a new network and has not received agent advertisements, it can initiate an **gent solicitation.** It can use the ICMP solicitation message to inform an agent that it needs assistance.

Mobile IP does not use a new packet type for agent solicitation; it uses the router solicitation packet of ICMP.

Registration

The second phase in mobile communication is **registration.** After a mobile host has moved to a foreign network and discovered the foreign agent, it must register. There are four aspects of registration:

1. The mobile host must register itself with the foreign agent.
2. The mobile host must register itself with its home agent. This is normally done by the foreign agent on behalf of the mobile host.
3. The mobile host must renew registration if it has expired.
4. The mobile host must cancel its registration (deregistration) when it returns home.

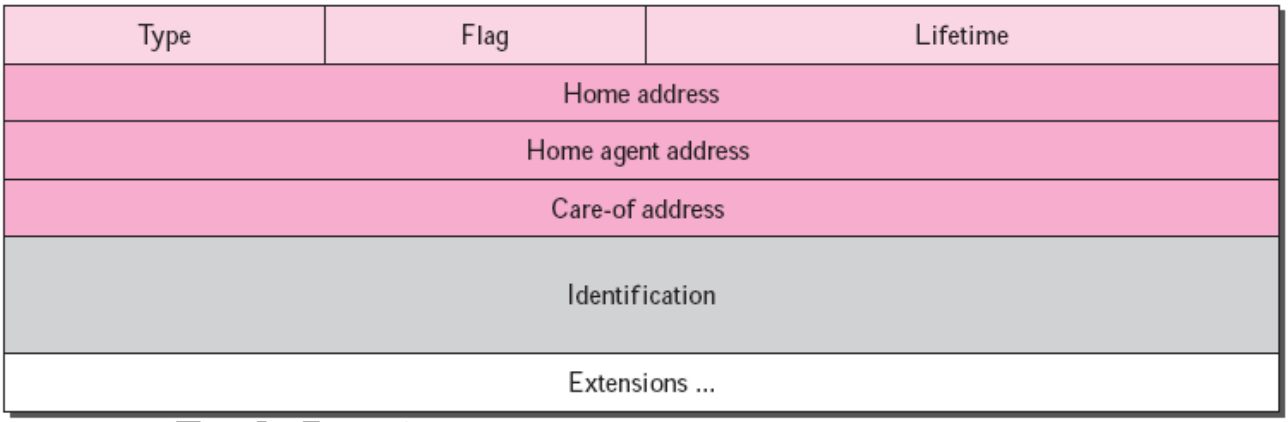
Request and Reply

To register with the foreign agent and the home agent, the mobile host uses a **registration request** and a **registration reply** as shown in Figure

Registration Request

A registration request is sent from the mobile host to the foreign agent to register its care-of address and also to announce its home address and home agent address. The foreign agent, after receiving and registering the request, relays the message to the home agent. Note that the home agent now knows the address of the foreign agent because the IP packet that is used for relaying has the IP address of the foreign agent as the source address. Figure shows the format of the registration request.

Registration request format



The field descriptions are as follows:

- **Type.** The 8-bit type field defines the type of the message. For a request message the value of this field is 1.
- **Flag.** The 8-bit flag field defines forwarding information. The value of each bit can be set or unset. The meaning of each bit is given in Table

Bit	Meaning
0	Mobile host requests that home agent retain its prior care-of address.
1	Mobile host requests that home agent tunnel any broadcast message.

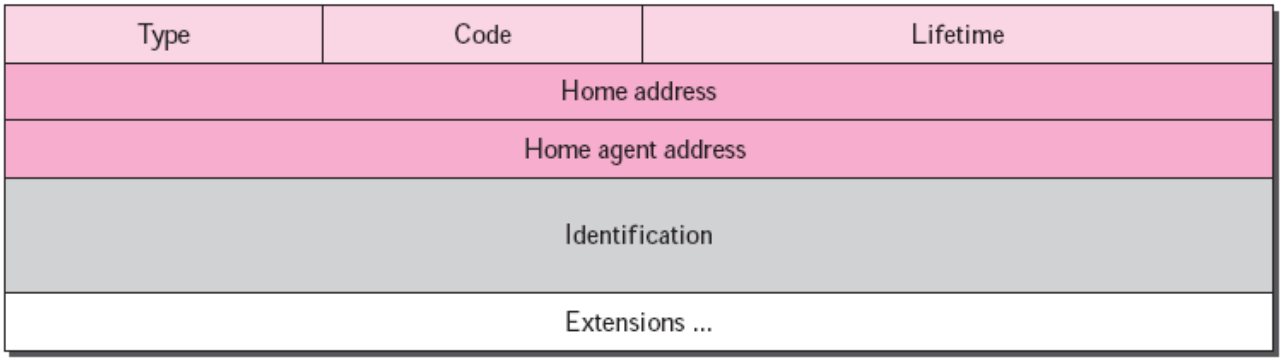
2	Mobile host is using colocated care-of address.
3	Mobile host requests that home agent use minimal encapsulation.
4	Mobile host requests generic routing encapsulation (GRE).
5	Mobile host requests header compression.
6–7	Reserved bits.

- **Lifetime.** This field defines the number of seconds the registration is valid. If the field is a string of 0s, the request message is asking for deregistration. If the field is a string of 1s, the lifetime is infinite.
- **Home address.** This field contains the permanent (first) address of the mobile host.
- **Home agent address.** This field contains the address of the home agent.
- **Care-of address.** This field is the temporary (second) address of the mobile host.
- **Identification.** This field contains a 64-bit number that is inserted into the request by the mobile host and repeated in the reply message. It matches a request with a reply.
- **Extensions.** Variable length extensions are used for authentication. They allow a home agent to authenticate the mobile agent..

Registration Reply

A registration reply is sent from the home agent to the foreign agent and then relayed to the mobile host. The reply confirms or denies the registration Request. Figure shows the format of the registration reply.

The value of the type field is 3. The code field replaces the flag field and shows the result of the registration request (acceptance or denial). The care-of address field is not needed.



Encapsulation

Registration messages are encapsulated in a UDP user datagram. An agent uses the well-known port 434; a mobile host uses an ephemeral port.

Data Transfer

After agent discovery and registration, a mobile host can communicate with a remote host.

From Remote Host to Home Agent

When a remote host wants to send a packet to the mobile host, it uses its address as the source address and the home address of the mobile host as the destination address. In other words, the remote host sends a packet as though the mobile host is at its home network. The packet, however, is intercepted by the home agent, which pretends it is the mobile host. This is done using the proxy ARP technique

From Home Agent to Foreign Agent

After receiving the packet, the home agent sends the packet to the foreign agent using the tunneling concept

From Foreign Agent to Mobile Host

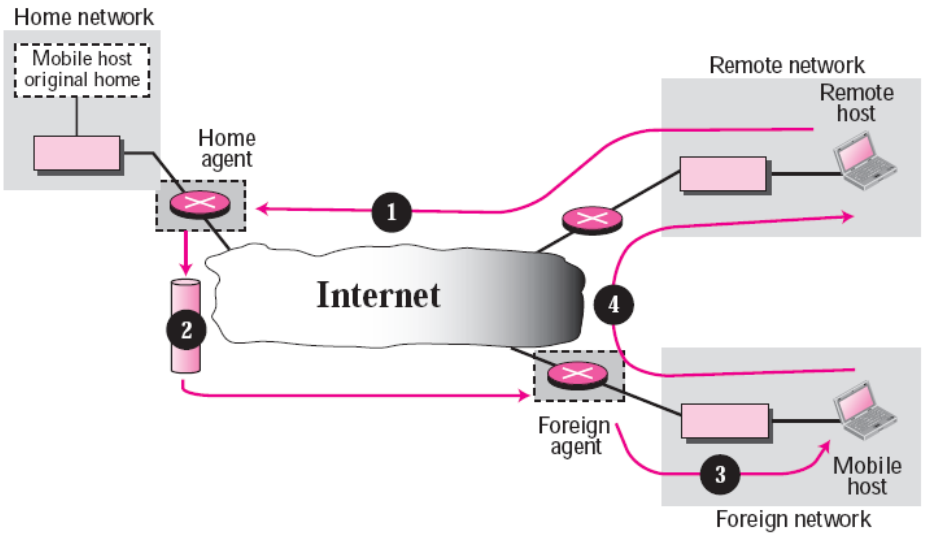
When the foreign agent receives the packet, it removes the original packet. However, since the destination address is the home address of the mobile host, the foreign agent consults a registry table to find the care-of address of the mobile host. (Otherwise, the packet would just be sent back to the home network.) The packet is then sent to the care-of address.

From Mobile Host to Remote Host

When a mobile host wants to send a packet to a remote host (for example, a response to the packet it has received), it sends as it does normally. The mobile host prepares a packet with its home address as the source, and the address of the remote host as the destination. Although the packet comes from the foreign network, it has the home address of the mobile host.

Transparency

In this data transfer process, the remote host is unaware of any movement by the mobile host. The remote host sends packets using the home address of the mobile host as the destination address; it receives packets that have the home address of the mobile host as the mobile host. This is done using the proxy ARP technique

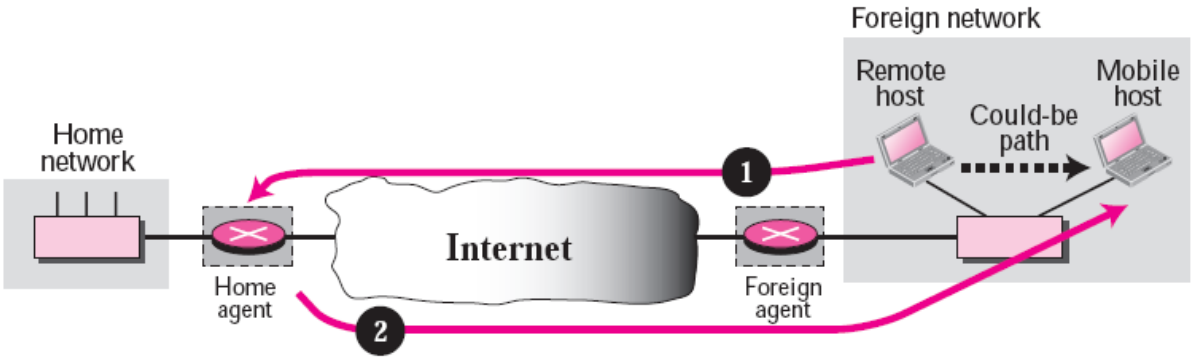


INEFFICIENCY IN MOBILE IP

Communication involving mobile IP can be inefficient. The inefficiency can be severe or moderate. The severe case is called *double crossing* or *2X* . The moderate case is called *triangle routing* or *dog-leg routing*

Double Crossing

Double crossing occurs when a remote host communicates with a mobile host that has moved to the same network (or site) as the remote host



Triangle Routing

Triangle routing, the less severe case, occurs when the remote host communicates with a mobile host that is not attached to the same network (or site) as the mobile host.

Solution

One solution to inefficiency is for the remote host to bind the care-of address to the home address of a mobile host.

UNICAST ROUTING PROTOCOLS –

- A routing table can be either static or dynamic. A *static table* is one with manual entries.A *dynamic table*, on the other hand, is one that is updated automatically when there is a change somewhere in the internet.
- Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a router is down, and they need to be updated whenever a better route has been found.
- Routing protocols have been created in response to the demand for dynamic routing tables. A routing protocol is a combination of rules and procedures that lets routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhood.

Optimization –

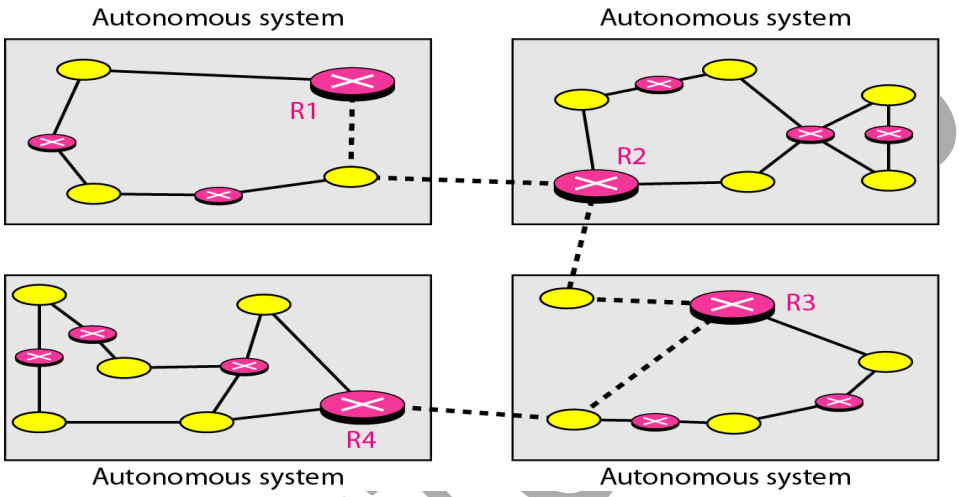
A router receives a packet from a network and passes it to another network. A router is usually attached to several networks. When it receives a packet, to which network should it pass the packet? The decision is based on optimization: Which of the available pathways is the optimum pathway(shortest or nearest path)?

- One approach is to assign a cost for passing through a network is called as metric. However, the metric assigned to each network depends on the type of protocol.
- Some simple protocols, such as the Routing Information Protocol (RIP), treat all networks as equals. The cost of passing through a network is the same; it is one hop count. So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts.
- Other protocols, such as Open Shortest Path First (OSPF), allow the administrator to assign a cost for passing through a network based on the type of service required. A route through a network can have different costs (metrics). Routers use routing tables to help decide the bestroute. OSPF protocol allows each router to have several routing tables based on the required type of service.
- Other protocols define the metric in a totally different way. In the Border Gateway Protocol (BGP), the criterion is the policy, which can be set by the administrator. The policy defines what paths should be chosen.

Intra- and Interdomain Routing –

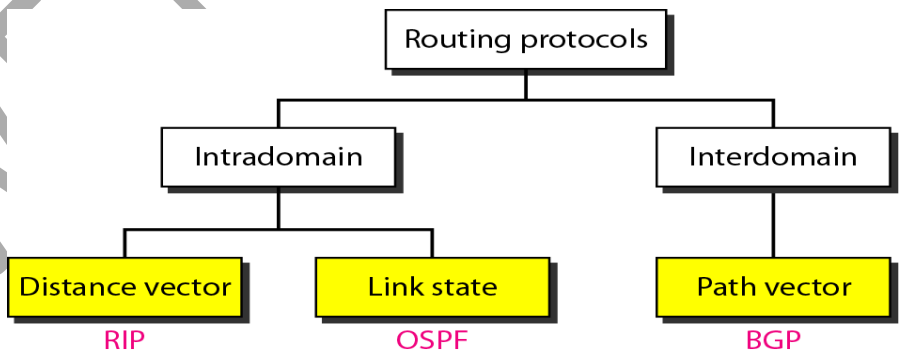
- An internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems.

- An autonomous system (AS) is a group of networks and routers under the authority of a single administration. Routing inside an autonomous system is referred to as Intradomain Routing.
- Routing between autonomous systems is referred to as Interdomain Routing. Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system. However, only one interdomain routing protocol handles routing between autonomous systems as shown in figure –



- Several Intradomain and Interdomain routing protocols are in use. In this section, Two Intradomain Routing protocols: Distance Vector and Link State. One Interdomain Routing protocol: Path Vector as shown in figure –

1. Routing Information Protocol (RIP) is an implementation of the distance vector protocol.
2. Open Shortest Path First (OSPF) is an implementation of the link state protocol.
3. Border Gateway Protocol (BGP) is an implementation of the path vector protocol.



Distance Vector Routing –

- In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, each node maintains a vector (table) of minimum distances to every node.
- The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing). In Figure(a), we show a system of five nodes with their corresponding tables.

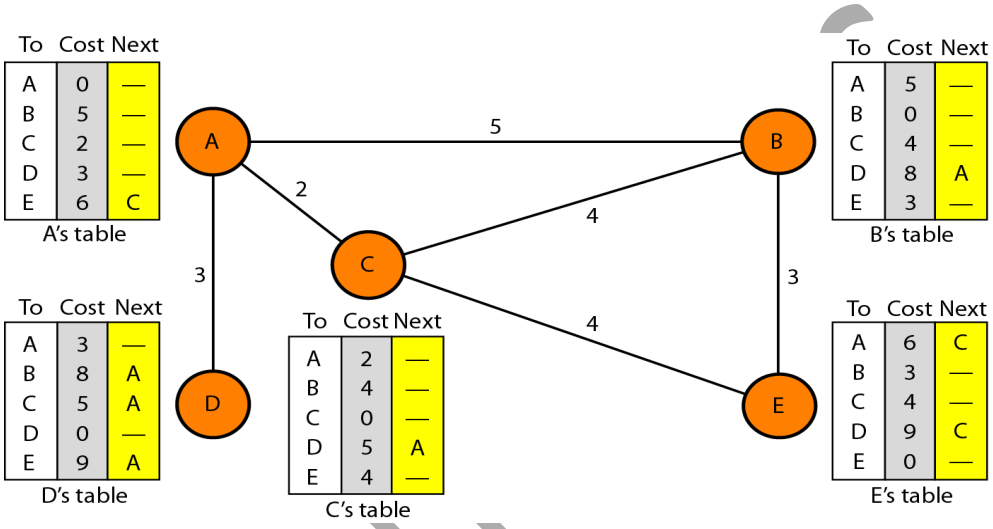


Figure (a) – Distance Vector Routing Tables

- The table for node A shows how we can reach any node from this node. For example,our least cost to reach node E is 6. The route passes through C.

Initialization –

- The tables in Figure(a) are stable; each node knows how to reach any other node and the cost. At the beginning, however, this is not the case. Each node can know only the distance between itself and its immediate neighbors, those directly connected to it.
- So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors. Figure(b) shows the initial tables for each node. The distance for any entry that is not a neighbor is marked as infinite (unreachable).

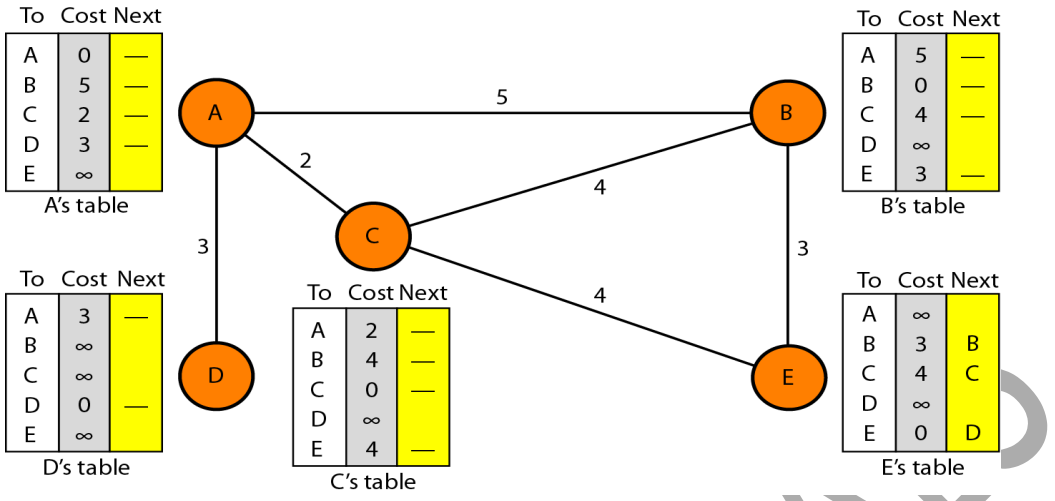


Figure – Initialization tables in Distance Vector Routing

Sharing –

- The whole idea of distance vector routing is the sharing of information between neighbors. Although node A does not know about node E, node C does. So if node C shares its routing table with A, node A can also know how to reach node E.
- On the other hand,node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.

Updating –

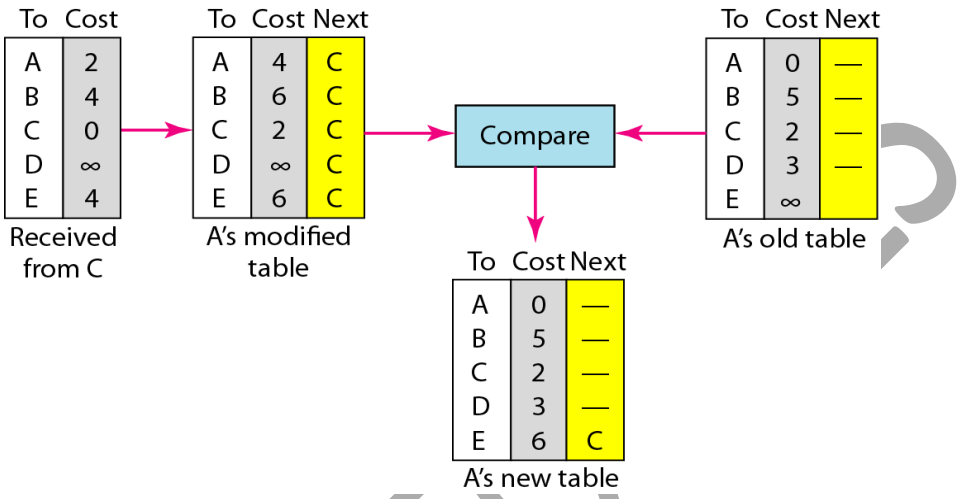
- When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is *x* mi, and the distance between A and C is *y* mi, then the distance between A and that destination, via C, is *x* + *y* mi.

2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.

3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is a tie, the old one is kept. If the next-node entry is the same, the receiving node chooses the new row.

- Figure(c) shows how node A updates its routing table after receiving the partial table from node C.



RIP –

The Routing Information Protocol (RIP) is an intradomain routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with routers and networks (links). The routers have routing tables; networks do not.
2. The destination in a routing table is a network, which means the first column defines a network address.
3. The metric used by RIP is very simple; the distance is defined as the number of links (networks) to reach the destination. For this reason, the metric in RIP is called a hop count.
4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
5. The next-node column defines the address of the router to which the packet is to be sent to reach its destination.

- ✓ Figure – 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.

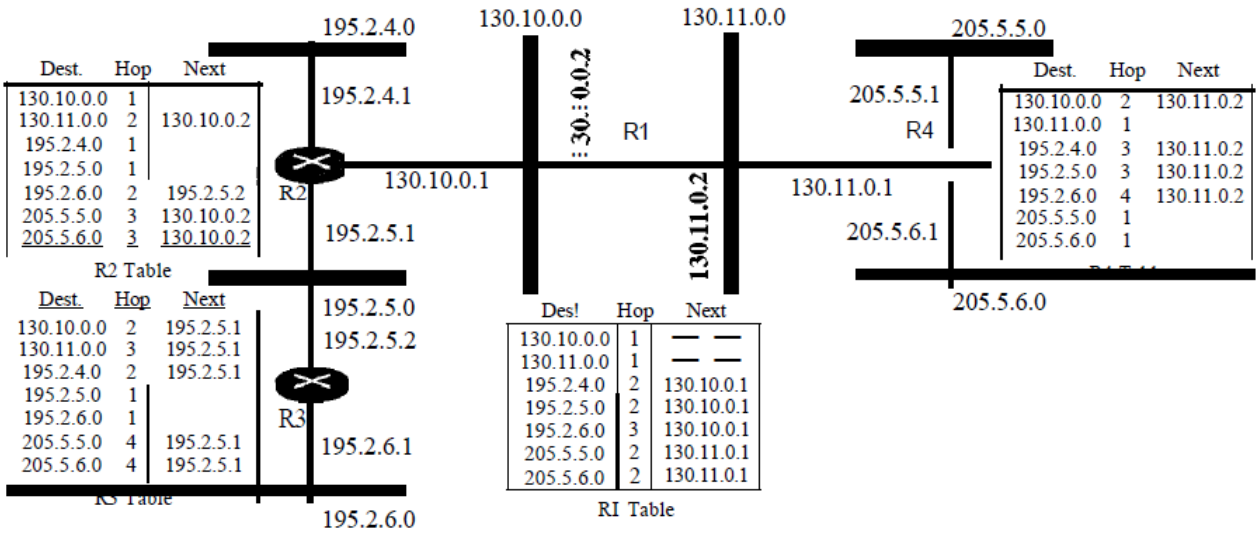


Figure – Example of domain using RIP

Link State Routing –

In link state routing, if each node in the domain has the entire topology of the domain – the list of nodes and links, how they are connected including the type, cost and condition of the links – the node can use dijkstra's algorithm to build a routing table as shown in Figure –

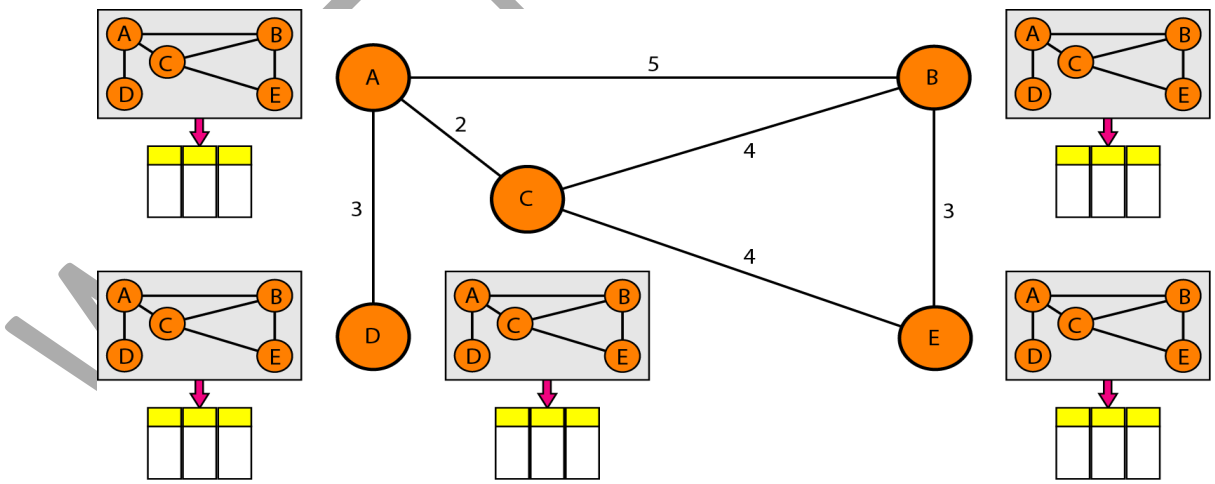


Figure (a) – Concept of Link State Routing

- The figure shows a simple domain with five nodes. Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology.

- This is analogous to a city map. While each person may have the same map, each needs to take a different route to reach her specific destination.
- The topology must be dynamic, representing the latest state of each node and each link. If there are changes in any point in the network (a link is down, for example), the topology must be updated for each node.
- In other words, the whole topology can be compiled from the partial knowledge of each node. Figure – (b) shows the same domain as in Figure – (a),indicating the part of the knowledge belonging to each node.

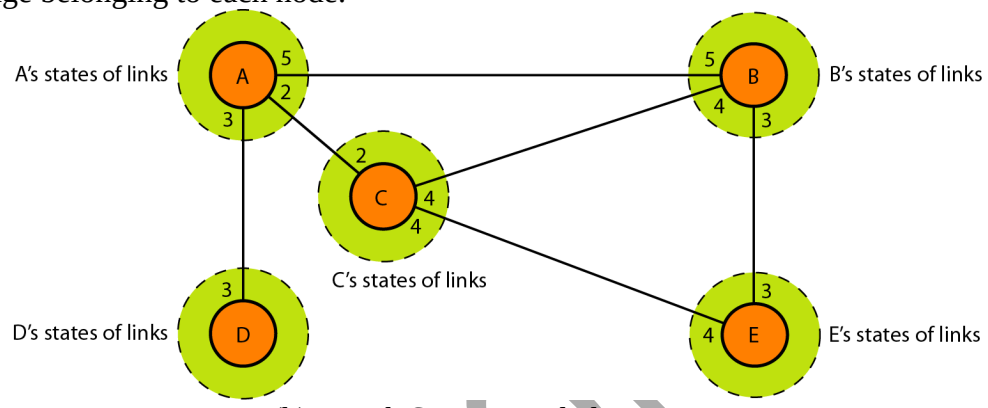


Figure (b) – Link State Knowledge

- Node A knows that it is connected to node B with metric 5, to node C with metric 2, and to node D with metric 3. Node C knows that it is connected to node A with metric 2, to node B with metric 4, and to node E with metric 4. Node D knows that it is connected only to node A with metric 3. And so on.

Building Routing Tables

In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding,** in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP) A link state packet can carry a large amount of information. For the moment, however, we assume that it carries a minimum amount of data: the node identity, the list of links, a sequence number, and age. The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from remaining in the domain for a long time. LSPs are generated on
two occasions:

1. When there is a change in the topology of the domain. Triggering of LSP dissemination is the main way of quickly informing any node in the domain to update its topology.
2. On a periodic basis. The period in this case is much longer compared to distance vector routing. As a matter of fact, there is no actual need for this type of LSP dissemination.

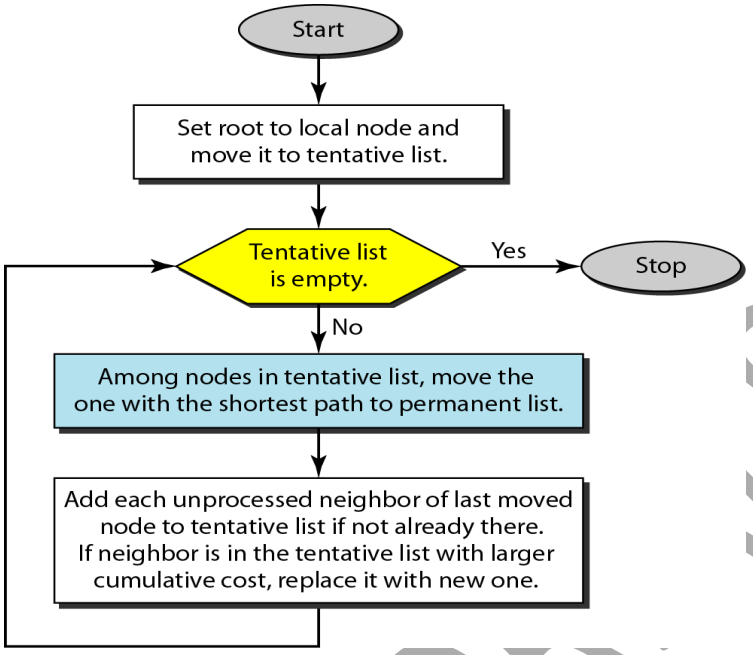
It is done to ensure that old information is removed from the domain. The timer set for periodic dissemination is normally in the range of 60 min or 2 hr based on the implementation. A longer period ensures that flooding does not create too much traffic on the network.

Flooding of LSPs – After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding and based on the following:

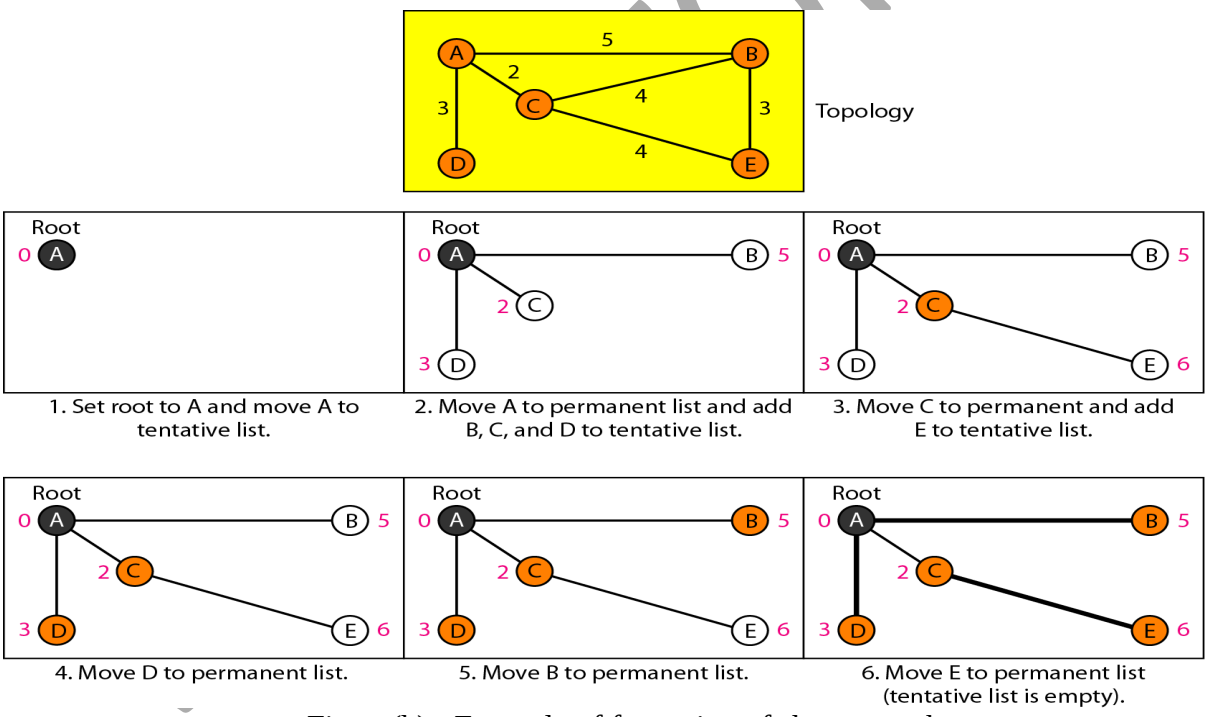
1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:
   a. It discards the old LSP and keeps the new one.
   b. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

Formation of Shortest Path Tree: Dijkstra Algorithm –

- After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.
- A tree is a graph of nodes and links; one node is called the root. All other nodes can be reached from the root through only one single route. A shortest path tree is a tree in which the path between the root and every other node is the shortest.
- The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: tentative and permanent.
- It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent. We can informally define the algorithm by using the flowchart in Figure(a) – Let us apply the algorithm to node A of our sample graph in Figure(b) . To find the shortest path in each step, we need the cumulative cost from the root to each node, which is shown next to the node.
- The following shows the steps. At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.



Figure(a) - Dijkstra algorithm



Figure(b) - Example of formation of shortest path tree

1. We make node A the root of the tree and move it to the tentative list. Our two lists are Permanent list: empty Tentative list: A(O)
2. Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to the pennanent list and add all neighbors ofA to the tentative list. Our new lists are Permanent list: A(O) Tentative list: B(5), C(2), D(3)
3. Node C has the shortest cumulative cost from all nodes in the tentative list. We move C to the permanent list. Node C has three neighbors, but node A is already processed, which

makes the unprocessed neighbors just B and E. However, B is already in the tentative list with a cumulative cost of 5. Node A could also reach node B through C with a cumulative cost of 6. Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it. Our new lists are Permanent list: A(O), e(2) Tentative list: B(5), 0(3), E(6)

4. Node D has the shortest cumulative cost of all the nodes in the tentative list. We move D to the permanent list. Node D has no unprocessed neighbor to be added to the tentative list. Our new lists are Permanent list: A(O), C(2), 0(3) Tentative list: B(5), E(6)
5. Node B has the shortest cumulative cost of all the nodes in the tentative list. We move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (this is just node E). However, E(6) is already in the list with a smaller cumulative cost. The cumulative cost to node E, as the neighbor of B, is 8. We keep node E(6) in the tentative list. Our new lists are Permanent list: A(O), B(5), C(2), 0(3) Tentative list: E(6)
6. Node E has the shortest cumulative cost from all nodes in the tentative list. We move E to the permanent list. Node E has no neighbor. Now the tentative list is empty. We stop; our shortest path tree is ready. The final lists are Permanent list: A(O), B(5), C(2), D(3), E(6) Tentative list: empty

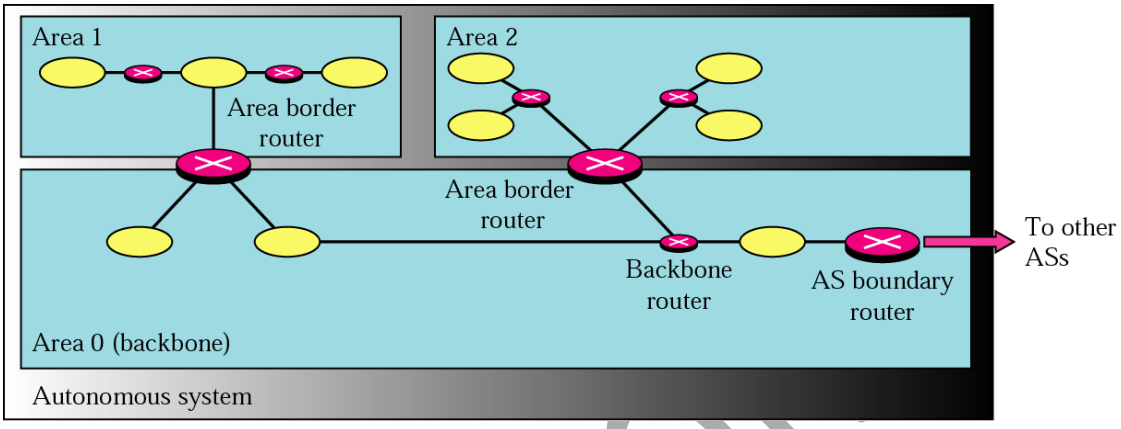
Calculation of Routing Table from Shortest Path Tree Each node uses the shortest path tree protocol to construct its routing table. The routing table shows the cost of reaching each node from the root. Table shows the routing table for node A.

Node	*Cost*	*Next Router*
A	0	-
B	5	-
C	2	-
D	3	-
E	6	C

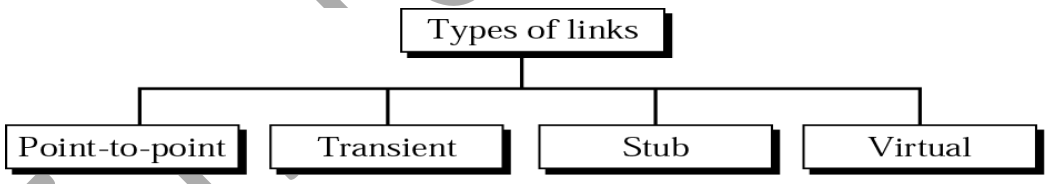
OSPF

- The Open Shortest Path First or OSPF protocol is an intradomain routing protocol based on link state routing. Its domain is also an autonomous system.
- Areas To handle routing efficiently and in a timely manner, OSPF divides an autonomous system into areas. An area is a collection of networks, hosts, and routers all contained within an autonomous system.
- An autonomous system can be divided into many different areas. All networks inside an area must be connected. Routers inside an area flood the area with routing information. At the border of an area, special routers called area border routers summarize the information about the area and send it to other areas.
- Among the areas inside an autonomous system is a special area called the backbone; all the areas inside an autonomous system must be connected to the backbone. In other words, the backbone serves as a primary area and the other areas as secondary areas. This does not mean that the routers within areas cannot be connected to each other, however. The routers inside the backbone are called the backbone routers.

- Note that a backbone router can also be an area border router. If, because of some problem, the connectivity between a backbone and an area is broken, a virtual link between routers must be created by an administrator to allow continuity of the functions of the backbone as the primary area.Each area has an area identification. The area identification of the backbone is zero. Figure – shows an autonomous system and its areas.

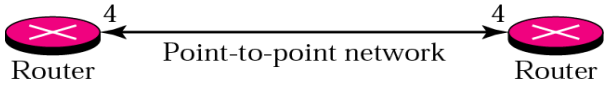


- Metric The OSPF protocol allows the administrator to assign a cost, called the metric, to each route. The metric can be based on a type of service (minimum delay, maximum throughput, and so on).
- Types of Links In OSPF terminology, a connection is called a link. Four types of links have been defined: point-to-point, transient, stub, and virtual as shown in figure –



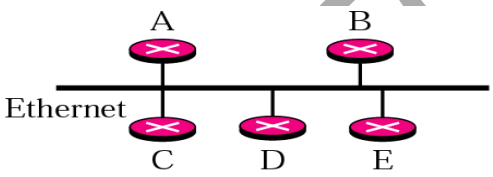
Point-to-point Link –

- A point-to-point link connects two routers without any other host or router in between. In other words, the purpose of the link (network) is just to connect the two routers. An example of this type of link is two routers connected by a telephone line or a T line.
- There is no need to assign a network address to this type of link. Graphically, the routers are represented by nodes, and the link is represented by a bidirectional edge connecting the nodes. The metrics, which are usually the same, are shown at the two ends, one for each direction. In other words, each router has only one neighbor at the other side of the link as shown in figure –

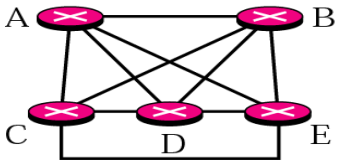


Transient Link –

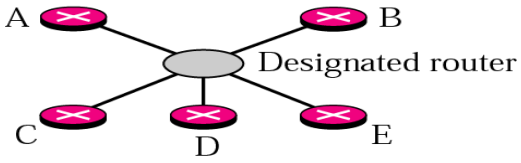
- A transient link is a network with several routers attached to it. The data can enter through any of the routers and leave through any router. All LANs and some WANs with two or more routers are of this type.
- In this case, each router has many neighbors. For example, consider the Ethernet in Figure – (a) . Router A has routers B, C, D, and E as neighbors. Router B has routers A, C, D, and E as neighbors and so on. If we want to show the neighborhood relationship in this situation, we have the graph shown in Figure – (b).
- This is neither efficient nor realistic. It is not efficient because each router needs to advertise the neighborhood to four other routers, for a total of 20 advertisements. It is not realistic because there is no single network (link) between each pair of routers; there is only one network that serves as a crossroad between all five routers.
- To show that each router is connected to every other router through one single network, the network itself is represented by a node. One of the routers in the network takes this responsibility. It is assigned a dual purpose; it is a true router and a designated router.
- We can use the topology shown in Figure –(c) to show the connections of a transient network.



a. Transient network



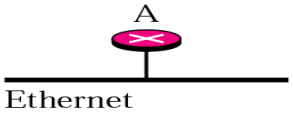
b. Unrealistic representation



c. Realistic representation

Stub Link –

- A **stub link** is a network that is connected to only one router. The data packets enter the network through this single router and leave the network through this same router.
- This is a special case of the transient network. We can show this situation using the router as a node and using the designated router for the network. However, the link is only one-directional, from the router to the network as shown in figure –



a. Stub network

b. Representation

Virtual Link –

When the link between two routers is broken, the administration may create a **virtual link** between them, using a longer path that probably goes through several routers.

Path Vector Routing –

- Distance vector and link state routing are both intradomain routing protocols. They can be used inside an autonomous system, but not between autonomous systems. These two protocols are not suitable for interdomain routing mostly because of scalability.
- In path vector routing, we assume that there is one node in each autonomous system that acts on behalf of the entire autonomous system is called as speaker node. The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs.
- The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other. A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems.

Initialization –

- At the beginning, each speaker node can know only the reachability of nodes inside its autonomous system. Figure – shows the initial tables for each speaker node in a system made of four ASs.

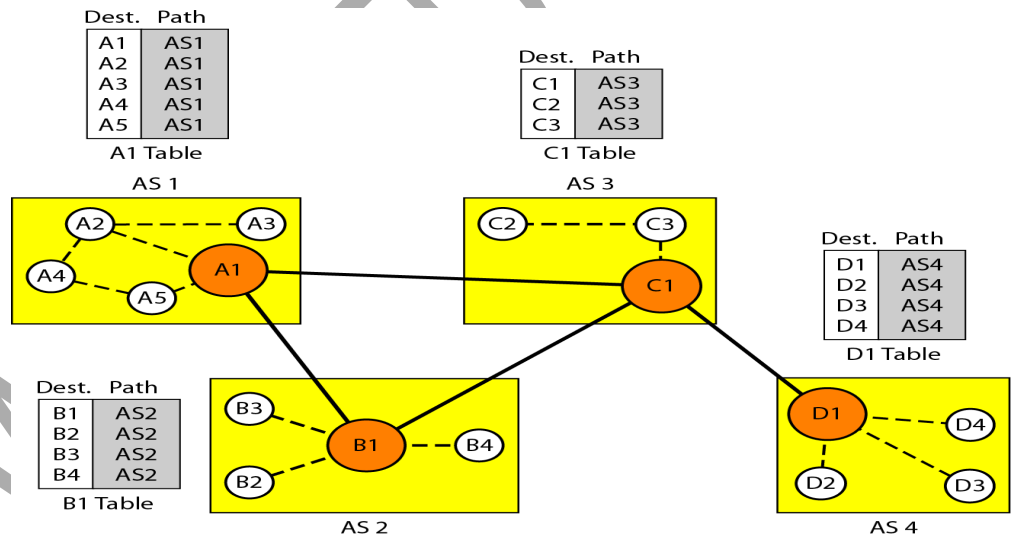


Figure – initial routing tables in path vector routing

- Node Al is the speaker node for ASl, Bl for AS2, Cl for AS3, and Dl for AS4. Node Al creates an initial table that shows Al to A5 are located in ASI and can be reached through it. Node Bl advertises that Bl to B4 are located in AS2 and can be reached through Bl. And so on.

Sharing –

- In path vector routing, a speaker in an autonomous system shares its table with immediate neighbors. In Figure – node Al shares its table with nodes Bl and Cl. Node Cl shares its table with nodes Dl, Bl, andAl. Node Bl shares its table with Cl andAl. Node Dl shares its table with Cl.

Updating –

- When a speaker node receives a two-column table from a neighbor, it updates its own table by adding the nodes that are not in its routing table and adding it own autonomous system and the autonomous system that sent the table.

Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. We can divide autonomous systems into three categories: stub, multihomed, and transit.
Stub AS –

- A stub AS has only one connection to another AS. The interdomain data traffic in a stub AS can be either created or terminated in the AS. The hosts in the AS can send data traffic to other ASs. The hosts in the AS can receive data coming from hosts in other ASs.
- A good example of a stub AS is a small corporation or a small local ISP.

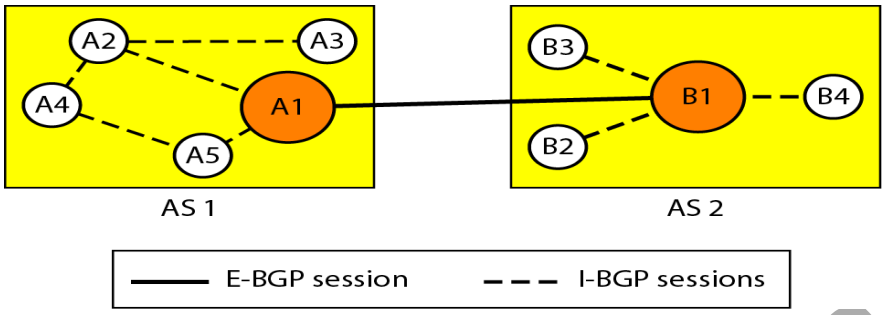
Multihomed AS –

- A multihomed AS has more than one connection to other ASs, but it is still only a source or sink for data traffic. It can receive data traffic from more than one AS. It can send data traffic to more than one AS, but there is no transient traffic.
- It does not allow data coming from one AS and going to another AS to pass through. A good example of a multihomed AS is a large corporation that is connected to more than one regional or national AS that does not allow transient traffic.

Transit AS –

- A transit AS is a multihomed AS that also allows transient traffic. Good examples of transit ASs are national and international ISPs

BGP can have two types of sessions: External BGP (E-BGP) and Internal BGP (I-BGP) sessions. The E - BGP session is used to exchange information between two speaker nodes belonging to two different
autonomous systems. The I-BGP session, on the other hand, is used to exchange routing information between two routers inside an autonomous system as shown in figure –

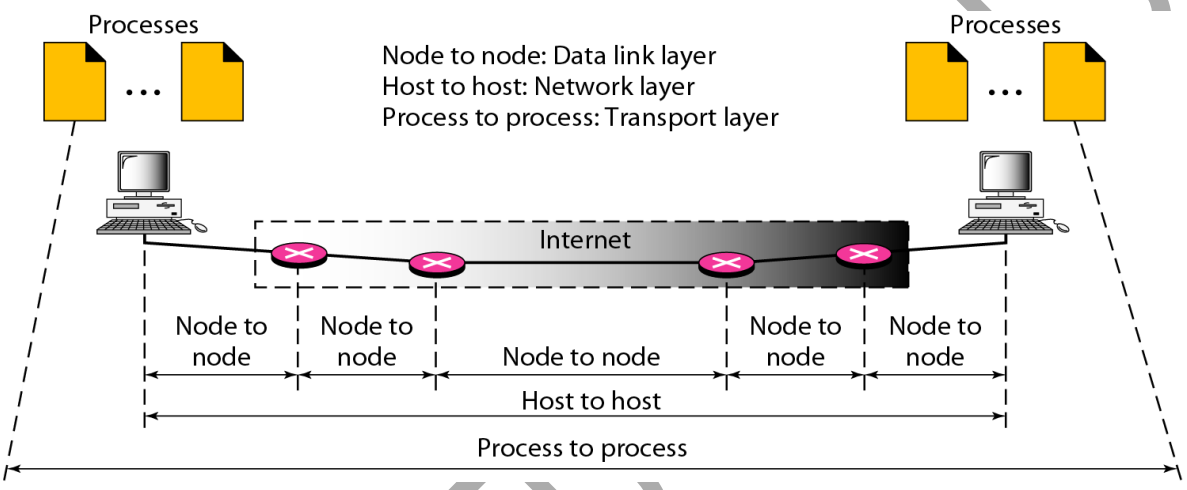


Unit III

Transport Layer –

PROCESS-TO-PROCESS DELIVERY

- The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, Figure shows these three types of deliveries and their domains.



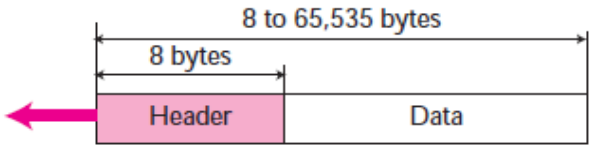
- The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called *node-to-node delivery*. The network layer is responsible for delivery of datagrams between two hosts. This is called *host-to-host delivery*.

USER DATAGRAM PROTOCOL (UDP)

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.
- Also, it performs very limited error checking.
- UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

User Datagram –

UDP packets, called **user datagrams,** have a fixed-size header of 8 bytes. Figure – shows the format of a user datagram. The fields are as follows:



a. UDP user datagram

b. Header format

Source port number.

- This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
- If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Destination port number –

- This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number.
- If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Length –

- This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.
- So if we subtract the value of the second field from the first, we can deduce the length of a UDP datagram that is encapsulated in an IP datagram.
- UDP length = IP length – IP header's length

Checksum – This field is used to detect errors over the entire user datagram (header plus data).

UDP Services –

Process-to-Process Communication

UDP provides process-to-process communication using sockets, a combination of IP addresses and port numbers. Several port numbers used by UDP are shown in Table –

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Connectionless Services –

- UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- The user datagrams are not numbered. Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.

Flow and Error Control –

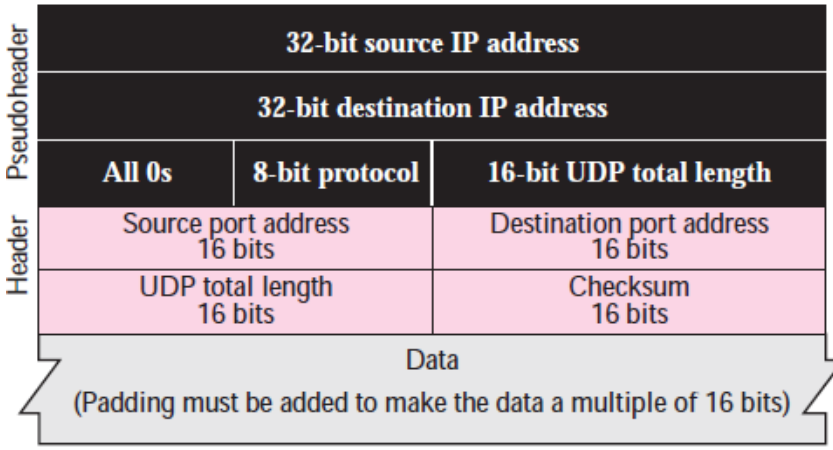
- UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded.

Encapsulation and Decapsulation –

- To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Checksum –

UDP checksum calculation is different from the one for IP. Here the checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.
The **pseudoheader** is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s (see Figure).

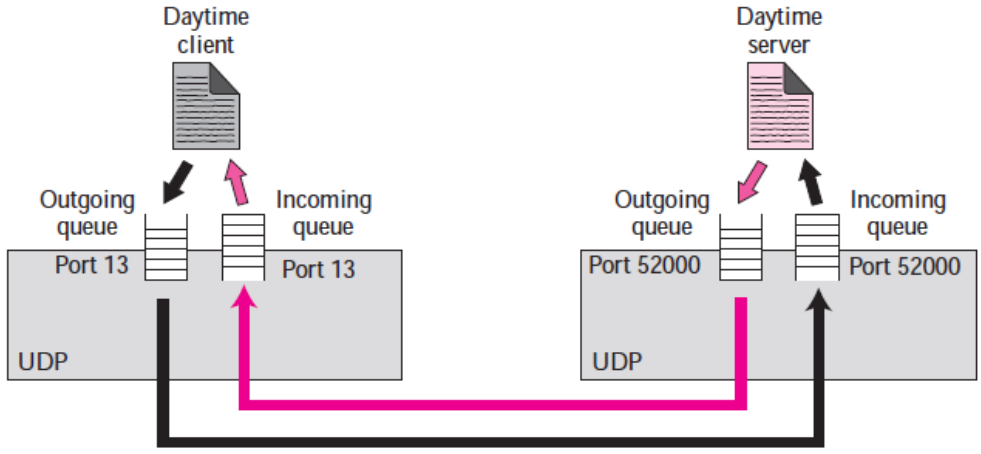


- If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.
- The protocol field is added to ensure that the packet belongs to UDP, and not to TCP. The value of the protocol field for UDP is 17. If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet. It is not delivered to the wrong protocol.

Congestion Control –

- Since UDP is a connectionless protocol, it does not provide congestion control. UDP assumes that the packets sent are small and sporadic, and cannot create congestion in the network. This assumption may or may not be true today when UDP is used for realtime transfer of audio and video.

Queuing –
In UDP, queues are associated with ports as shown in figure –



- At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue

associated with each process. Other implementations create only an incoming queue associated with each process.

- If a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue.
- The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow.
- If this happens, the operating system can ask the client process to wait before sending any more messages. When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a *port unreachable* message to the server.
- At the server site, the mechanism of creating queues is different. In its simplest form, a server asks for incoming and outgoing queues, using its well-known port, when it start running. The queues remain open as long as the server is running.
- When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the userdatagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client.

UDP Features –

Explain the advantages and disadvantages of UDP protocol –

Connectionless Service –

- Each UDP packet is independent from other packets sent by the same application program. This feature can be considered as an advantage or disadvantage depending on the application requirement. It is an advantage if, for example, a client application needs to send a short request to a server and to receive a short response. If the request and response can each fit in one single user datagram, a connectionless service may be preferable. The overhead to establish and close a connection may be significant in this case.
- The connectionless service provides less delay; the connection-oriented service creates more delay. If delay is an important issue for the application, the connectionless service is preferred.

Lack of Error Control –

- UDP does not provide error control; it provides an unreliable service. Most applications expect reliable service from a transport-layer protocol. Although a reliable service is desirable, it may have some side effects that are not acceptable to some applications.
- When a transport layer provides reliable services, if a part of the message is lost or corrupted, it needs to be resent. This means that the receiving transport layer cannot

deliver that part to the application immediately; there is an uneven delay between different parts of the message delivered to the application layer.

Lack of Congestion Control –

- UDP does not provide congestion control. However, UDP does not create additional traffic in an error-prone network. TCP may resend a packet several times and thus contribute to the creation of congestion or worsen a congested situation. Therefore, in some cases, lack of error control in UDP can be considered an advantage when congestion is a big issue.

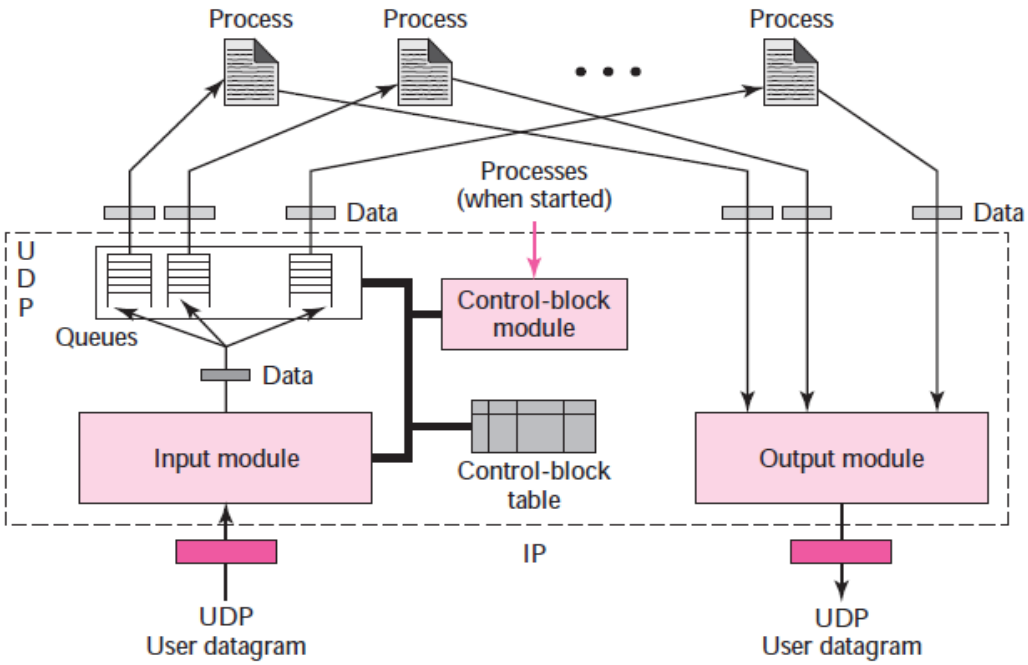
Typical Applications –

The following shows some typical applications that can benefit more from the services of UDP than from those of TCP.

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
- UDP is suitable for a process with internal flow and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP).
- UDP is normally used for real-time applications that cannot tolerate uneven delay between sections of a received message.

UDP PACKAGE –

The UDP package involves five components: a control-block table, input queues, a control-block module, an input module, and an output module. Figure – shows these five components and their interactions.



Control-Block Table – In our package, UDP has a control-block table to keep track of the open ports. Each entry in this table has a minimum of four fields: the state, which can be FREE or IN-USE, the process ID, the port number, and the corresponding queue number.

Input Queues – Our UDP package uses a set of input queues, one for each process. In this design, we do not use output queues.

Control-Block Module – The control-block module is responsible for the management of the control-block table. When a process starts, it asks for a port number from the operating system. The operating system assigns well-known port numbers to servers and ephemeral port numbers to clients. The process passes the process ID and the port number to the control-block module to create an entry in the table for the process. The module does not create the queues. The field for queue number has a value of zero.

Input Module – The input module receives a user datagram from the IP. It searches the control-block table to find an entry having the same port number as this user datagram. If the entry is found, the module uses the information in the entry to enqueue the data. If the entry is not found, it generates an ICMP message.

Output Module – The output module is responsible for creating and sending user datagrams.

TCP –

TCP, like UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

TCP Services –

Process-to-Process Communication –

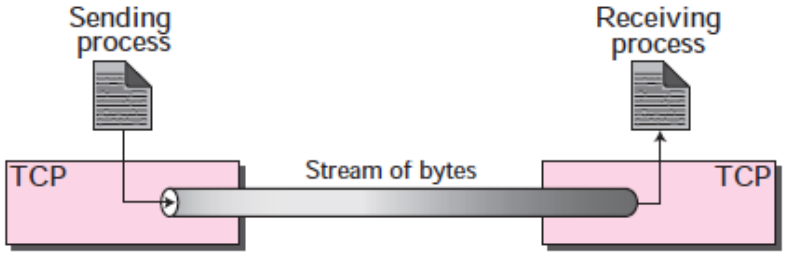
Like UDP, TCP provides process-to-process communication using port numbers. Table – lists some well-known port numbers used by TCP.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20 and 21	FTP	File Transfer Protocol (Data and Control)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol

Table – well known port number of TCP

Stream Delivery Service –

Its allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet.

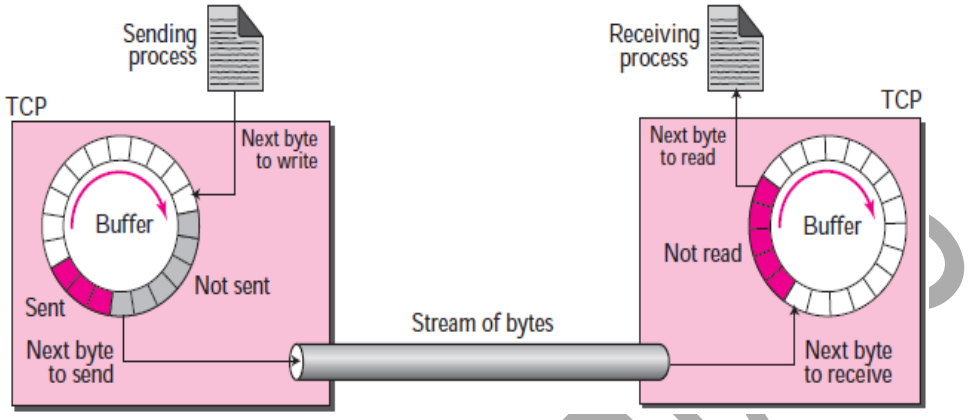


Sending and Receiving Buffers –

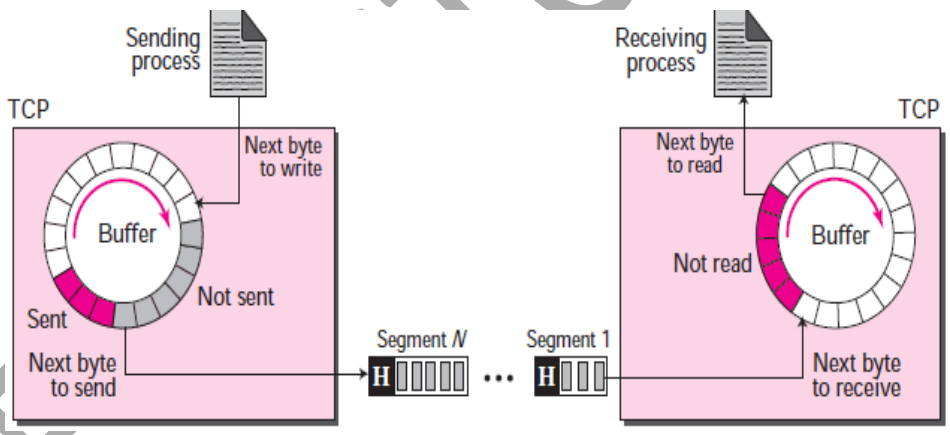
- The sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. (these buffers are also necessary for flow and

error control mechanisms used by TCP.) One way to implement a buffer is to use a circular array of I-byte locations as shown in Figure –

- For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation.



- At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process. The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP.



- At receiver side, the white area contains empty chambers to be filled by bytes received from the network. The colored sections contain received bytes that can be read by the receiving process.
- Segments, Although buffering handles the disparity between the speed of the producing and consuming processes, we need one more step before we can send data.
- The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment.
- Note that the segments are not necessarily the same size. In Figure – for simplicity, we show one segment carrying 3 bytes and the other carrying 5 bytes. In reality, segments carry hundreds, if not thousands, of bytes.

Full-Duplex Communication –

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions.

Multiplexing and Demultiplexing –

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver. However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

Connection-Oriented Service –

- TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:
    1. The two TCPs establish a connection between them.
    2. Data are exchanged in both directions.
    3. The connection is terminated.
- It is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent.
- Each may use a different path to reach the destination. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

Reliable Service –

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP Features –

Numbering System –

Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.

Byte Number –

TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.

Sequence Number –

After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

Acknowledgment Number –

The sequence number in each direction shows the number of the first byte carried by the segment. Each party also uses an acknowledgment number to confirm the bytes it has received. However, the acknowledgment number defines the number of the next byte that the party expects to receive.

Flow Control –

TCP, unlike UDP, provides *flow control*. The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

Error Control –

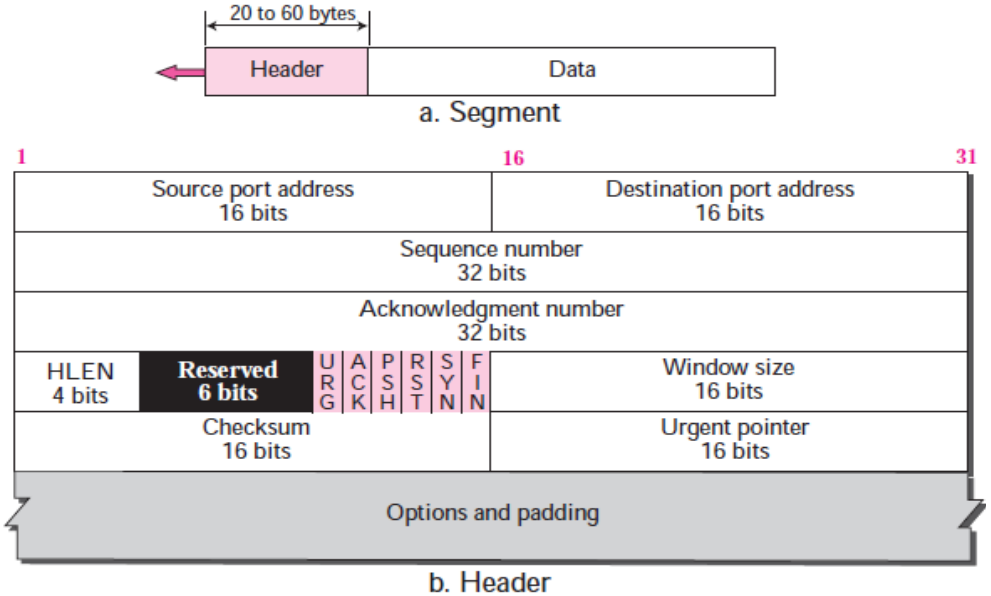
To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

Segment –

A packet in TCP is called a segment.

Format

The format of a segment is shown in Figure –



The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

The meaning and purpose of each field –

Source port address –

- This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.

Destination port address –

- This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.

Sequence number –

- This 32-bit field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.
- During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

Acknowledgment number –

- This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.
- If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x* + 1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

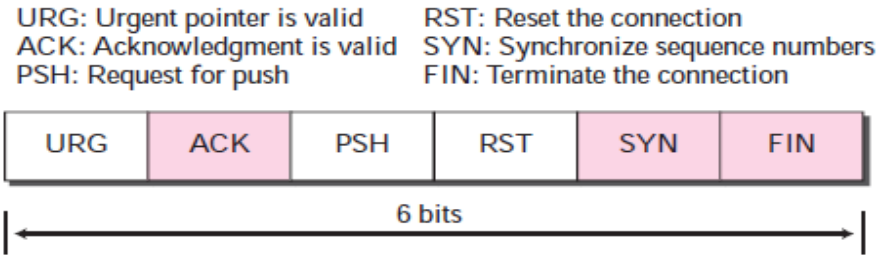
Header length –

- This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.

Reserved – This is a 6-bit field reserved for future use.

Control –

- This field defines 6 different control bits or flags as shown in Figure – .One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.



Window size –

- This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.

Checksum –

- This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP for detecting and correcting error(header and data).

Urgent pointer –

- This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

Options – There can be up to 40 bytes of optional information in the TCP header.

A TCP Connection –

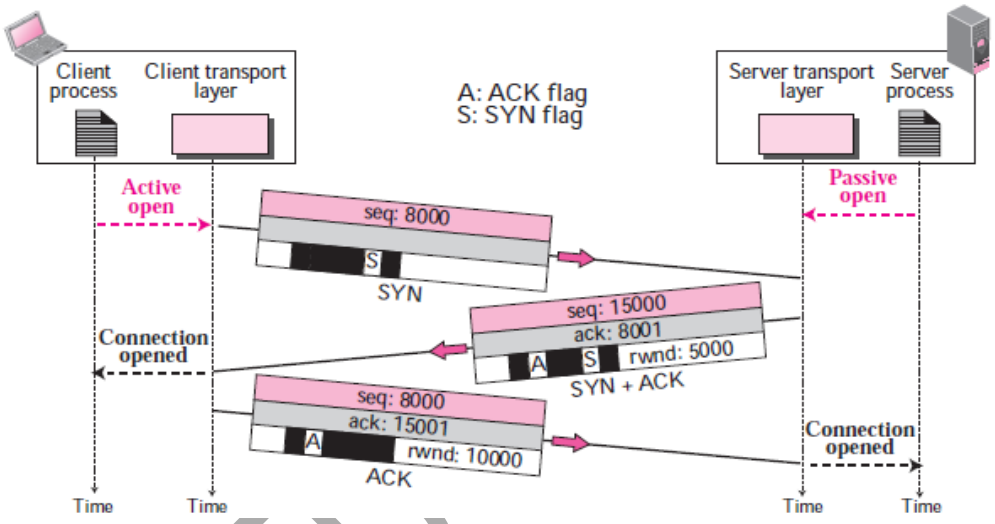
- TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path.
- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

Connection Establishment –

- TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.

Three-Way Handshaking –

- The connection establishment in TCP is called three way handshaking. In our example, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol.
- The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a *passive open.*
- The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure –



The three steps in this phase are as follows –

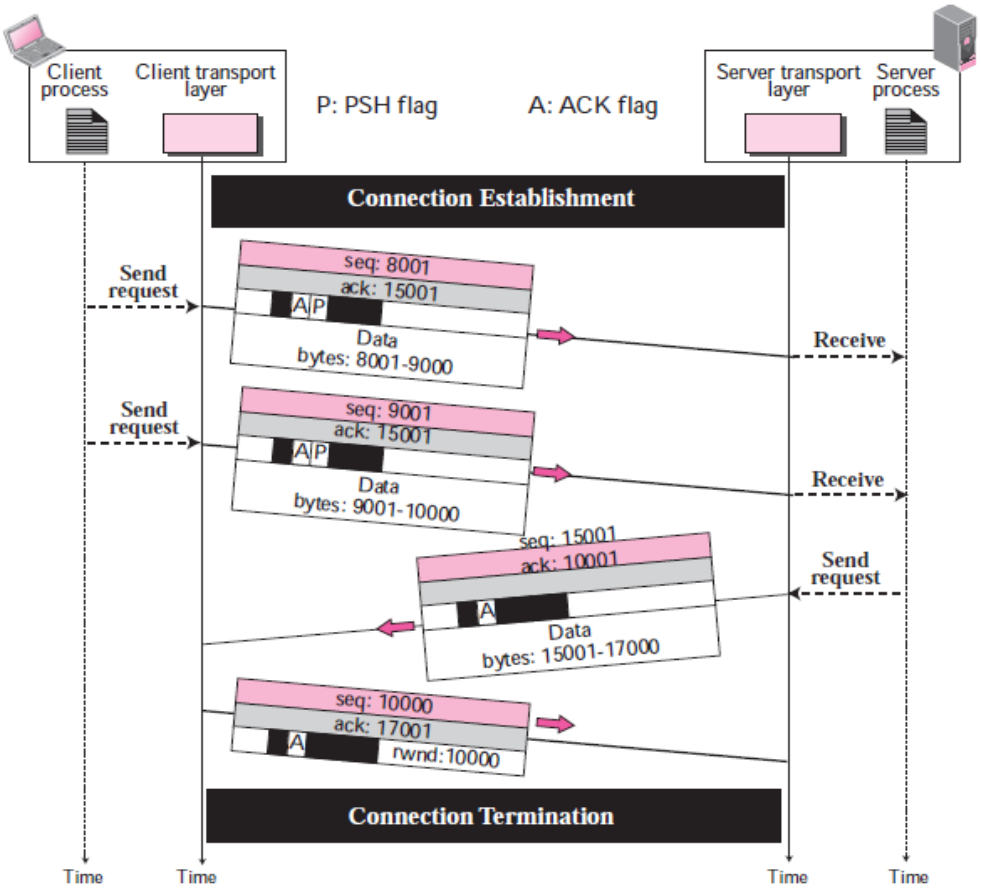
1. The client sends the first segment, a SYN segment, in which only the SYN flag is set.This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte. A SYN segment cannot carry data, but it consumes one sequence number.
2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number. A SYN +ACK segment cannot carry data, but does consume one sequence number.
3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers. An ACK segment, if carrying no data, consumes no sequence number.

Data Transfer –

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments.

Pushing Data –

- The sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size.
- The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP.



- The application program on one site wants to send a keystroke to the application at the other site and receive an immediate response. Delayed transmission and delayed delivery of data may not be acceptable by the application program. TCP can handle such a situation.
- The application program at the sending site can request a *push* operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately.

Urgent Data –

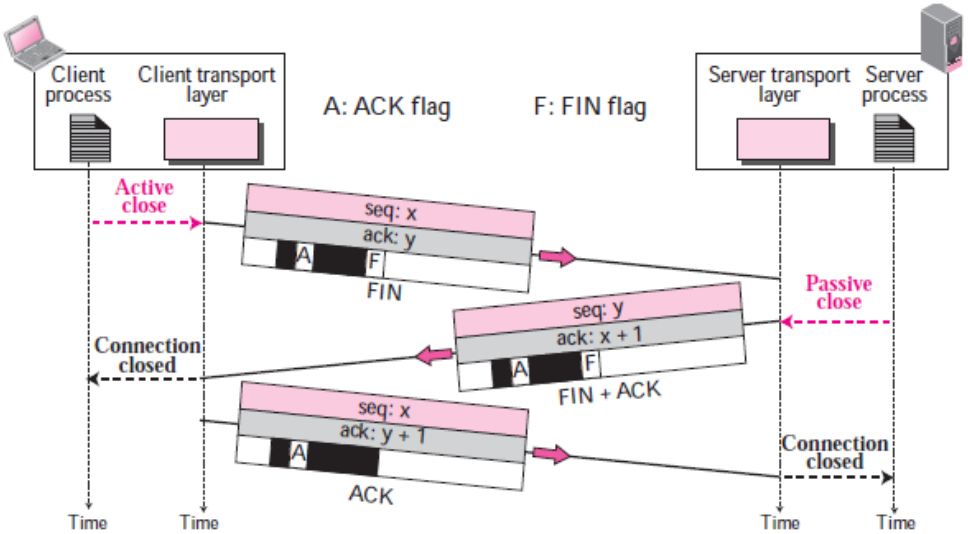
- TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream.
- However, an application program needs to send *urgent* bytes. This means that the sending application program wants a piece of data to be read out of order by the receiving application program.
- The solution is to send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer. The urgent pointer field in the header defines the end of the urgent data and the start of normal data.
- When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

Connection Termination –

Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.

Three-Way Handshaking –

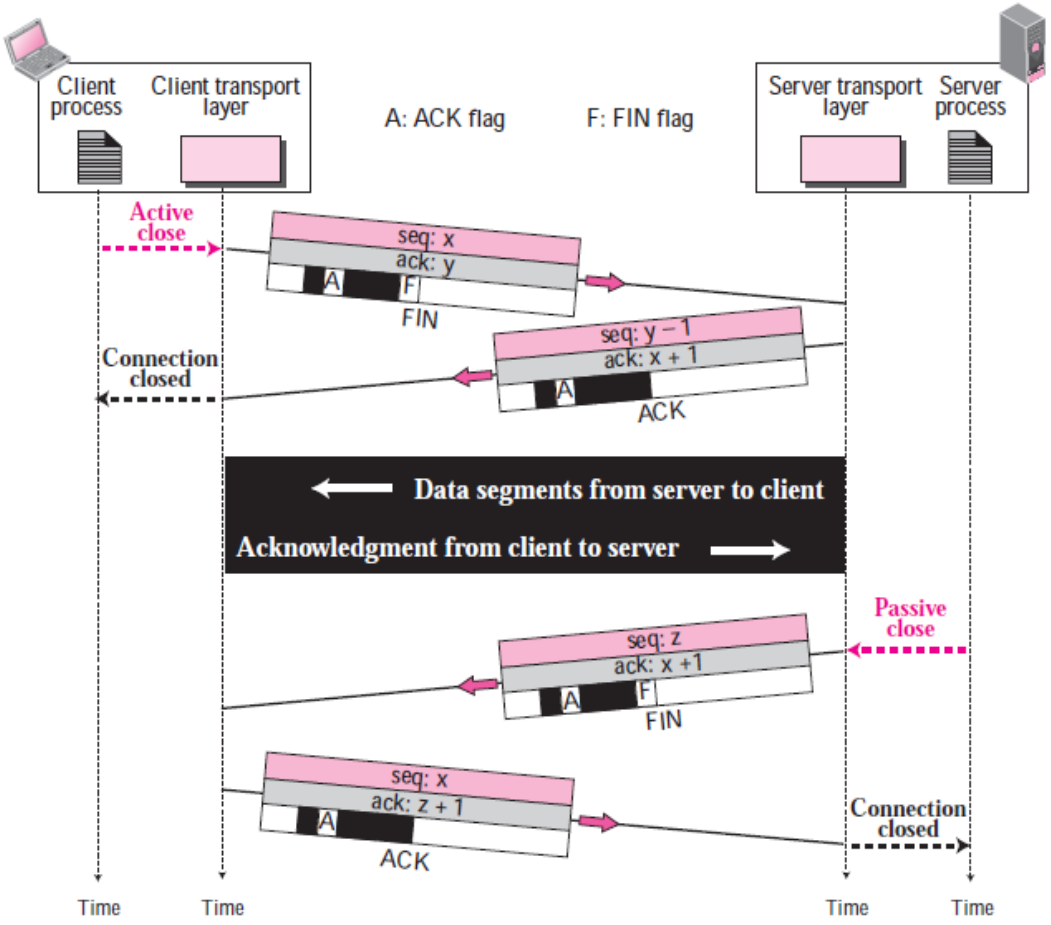
1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure – .If it is only a control segment, it consumes only one sequence number.



2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.
3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

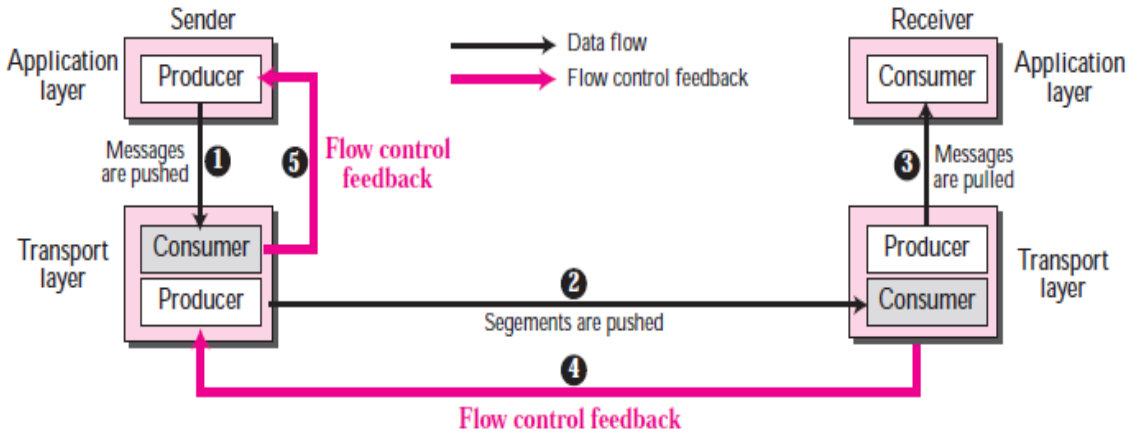
Half-Close –

- In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin.
- Figure – shows an example of a half-close. The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops. The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.
- After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server.



Flow Control –

- TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the *Go-Back-N* and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.
- Figure – shows the sliding window in TCP. The window spans a portion of the buffer containing bytes received from the process. The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment. The imaginary window has two walls: one left and one right.



- The window is *opened*, *closed*, or *shrunk*. These three activities, Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore. Shrinking the window means moving the right wall to the left.
- A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented.
- The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd)* or *congestion window (cwnd)*. The *receiver window* is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded. The congestion window is a value determined by the network to avoid congestion.

Error Control –

- TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments.
- Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

Checksum –

- Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

Acknowledgment –

- TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

Retransmission –

- The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted. A segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs.

Out-of-Order Segments –

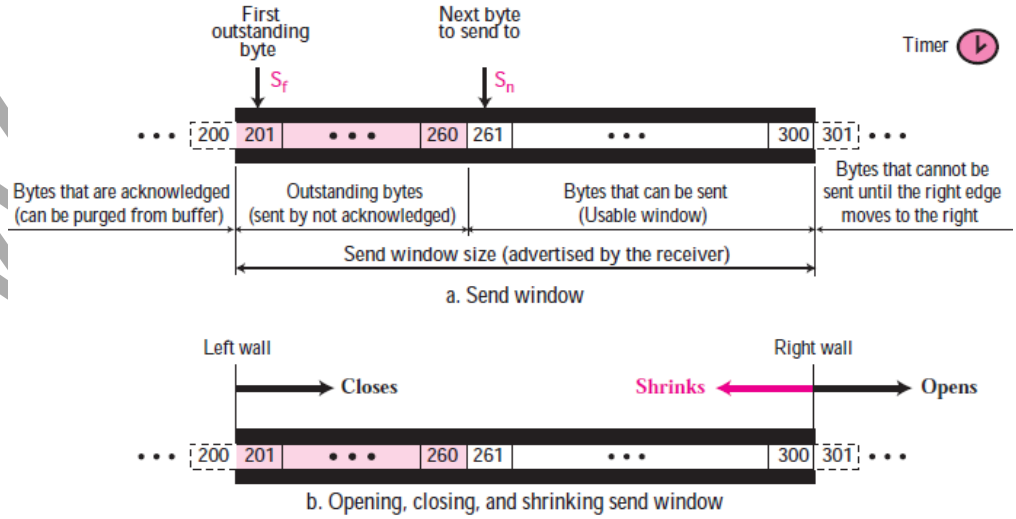
- When a segment is delayed, lost, or discarded, the segments following that segment arrive out of order. Originally, TCP was designed to discard all out-of-order segments, resulting in the retransmission of the missing segment and the following segments.

WINDOW IN TCP –

TCP uses two windows (send window and receive window) for each direction of data transfer, which means four windows for a bidirectional communication.

Send Window –

Figure – shows an example of a send window. The window we have used is of size 100 bytes (normally thousands of bytes), but later we see that the send window size is dictated by the receiver (flow control) and the congestion in the underlying network (congestion control). The figure shows how a send window *opens*, *closes*, or *shrinks*.

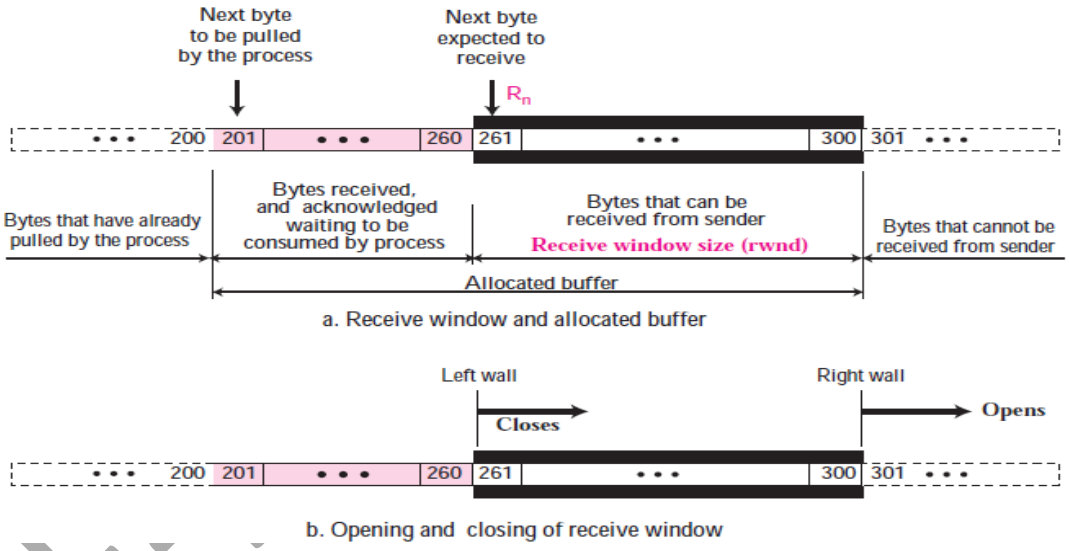


The send window in TCP is similar to one used with the Selective Repeat protocol, but with some differences:

- One difference is the nature of entities related to the window. The window in SR numbers pockets, but the window in the TCP numbers bytes. Although actual transmission in TCP occurs segment by segment, the variables that control the window are expressed in bytes.
- The second difference is that, in some implementations, TCP can store data received from the process and send them later, but we assume that the sending TCP is capable of sending segments of data as soon as it receives them from its process.
- Another difference is the number of timers. The theoretical Selective Repeat protocol may use several timers for each packet sent, but the TCP protocol uses only one timer. We later explain the use of this timer in error control.

Receive Window

Figure – shows an example of a receive window. The window we have used is of size 100 bytes (normally thousands of bytes). The figure also shows how the receive window opens and closes; in practice, the window should never shrink.



There are two differences between the receive window in TCP and the one we used for SR.

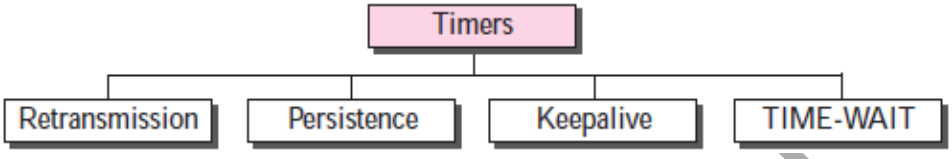
- The first difference is that TCP allows the receiving process to pull data at its own pace. This means that part of the allocated buffer at the receiver may be occupied by bytes that have been received and acknowledged, but are waiting to be pulled by the receiving process.
- The receive window size is then always smaller or equal to the buffer size, as shown in the above figure. The receiver window size determines the number of bytes that the receive window can accept from the sender before being overwhelmed (flow control). In other words, the receive window size, normally called *rwnd*, can be determined as:

  rwnd = buffer size – number of waiting bytes to be pulled

- The second difference is the way acknowledgments are used in the TCP protocol. Remember that an acknowledgement in SR is selective, defining the uncorrupted packets

that have been received. The major acknowledgment mechanism in TCP is a cumulative acknowledgment announcing the next expected byte to receive. The new versions of TCP, however,uses both cumulative and selective acknowledgements.

TCP TIMERS –



Retransmission Timer –

To retransmit lost segments, TCP employs one retransmission timer (for the whole connection period) that handles the retransmission time-out (RTO), the waiting time for an acknowledgment of a segment. We can define the following rules for the retransmission timer:

1. When TCP sends the segment in front of the sending queue, it starts the timer.
2. When the timer expires, TCP resends the first segment in front of the queue, and restarts the timer.
3. When a segment (or segments) are cumulatively acknowledged, the segment (or segments) are purged from the queue.
4. If the queue is empty, TCP stops the timer; otherwise, TCP restarts the timer.

Round-Trip Time (RTT) –

- To calculate the retransmission time-out (RTO), we first need to calculate the **roundtrip time (RTT).** However, calculating RTT in TCP is an involved process that we explain step by step with some examples.
- **Measured RTT** We need to find how long it takes to send a segment and receive an acknowledgment for it. This is the measured RTT. We need to remember that the segments and their acknowledgments do not have a one-to-one relationship; several segments may be acknowledged together.
- The measured round-trip time for a segment is the time required for the segment to reach the destination and be acknowledged, although the acknowledgment may include other segments.
- Note that in TCP, only one RTT measurement can be in progress at any time. This means that if an RTT measurement is started, no other measurement starts until the value of this RTT is finalized. We use the notation RTT_M to stand for measured RTT.
- **Smoothed RTT** The measured RTT, RTTM, is likely to change for each round trip. The fluctuation is so high in today's Internet that a single measurement alone cannot be used for retransmission time-out purposes. Most implementations use a smoothed RTT, called RTT_S, which is a weighted average of RTT_M and the previous RTT_S as shown below:

Initially	→	No value
After first measurement	→	$RTT_S = RTT_M$
After each measurement	→	$RTT_S = (1 - \alpha)\ RTT_S + \alpha \times RTT_M$

- The value of α is implementation-dependent, but it is normally set to 1/8. In other words, the new RTTS is calculated as 7/8 of the old RTTS and 1/8 of the current RTTM.
- RTT Deviation Most implementations do not just use RTTS; they also calculate the RTT deviation, called RTTD, based on the RTTS and RTTM using the following formulas:

Initially	→	No value
After first measurement	→	$RTT_D = RTT_M / 2$
After each measurement	→	$RTT_D = (1 - \beta)\ RTT_D + \beta \times \lvert RTT_S - RTT_M \rvert$

- The value of β is also implementation-dependent, but is usually is set to 1/4.

Retransmission Time-out (RTO) –

The value of RTO is based on the smoothed round-trip time and its deviation. Most implementations use the following formula to calculate the RTO:

Original	→	Initial value
After any measurement	→	$RTO = RTT_S + 4 \times RTT$

In other words, take the running smoothed average value of RTTS, and add four times the running smoothed average value of RTTD (normally a small value).

Persistence Timer –

- To deal with a zero-window-size advertisement, TCP needs another timer. If the receiving TCP announces a window size of zero, the sending TCP stops transmitting segments until the receiving TCP sends an ACK segment announcing a nonzero window size.
- This ACK segment can be lost. Remember that ACK segments are not acknowledged nor retransmitted in TCP. If this acknowledgment is lost, the receiving TCP thinks that it has done its job and waits for the sending TCP to send more segments. There is no retransmission timer for a segment containing only an acknowledgment.
- The sending TCP has not received an acknowledgment and waits for the other TCP to send an acknowledgment advertising the size of the window. Both TCPs might continue to wait for each other forever (a deadlock).
- To correct this deadlock, TCP uses a **persistence timer** for each connection. When the sending TCP receives an acknowledgment with a window size of zero, it starts a persistence timer. When the persistence timer goes off, the sending TCP sends a special segment called a probe.
- This segment contains only 1 byte of new data. It has a sequence number, but its sequence number is never acknowledged; it is even ignored in calculating the sequence number for the rest of the data. The probe causes the receiving TCP to resend the acknowledgment.

Keepalive Timer –

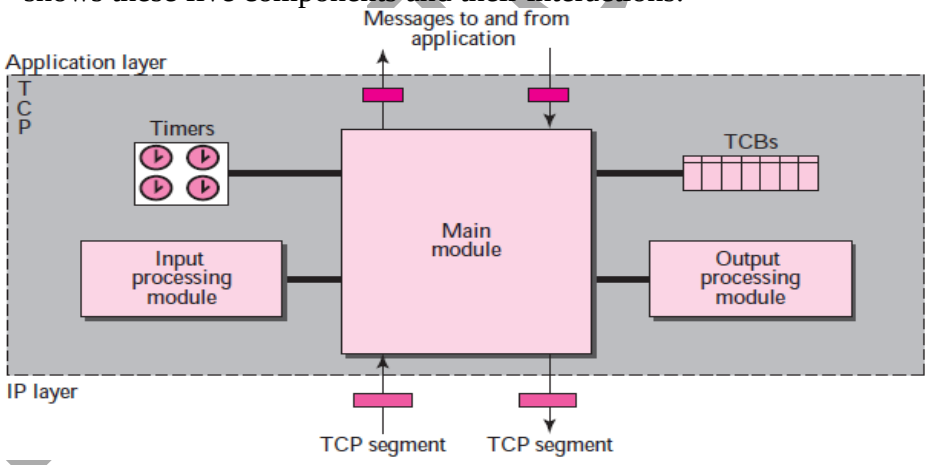
- A keepalive timer is used in some implementations to prevent a long idle connection between two TCPs. Suppose that a client opens a TCP connection to a server, transfers some data, and becomes silent.
- Perhaps the client has crashed. In this case, the connection remains open forever.
- To remedy this situation, most implementations equip a server with a keepalive timer. Each time the server hears from a client, it resets this timer. The time-out is usually 2 hours. If the server does not hear from the client after 2 hours, it sends a probe segment. If there is no response after 10 probes, each of which is 75 s apart, it assumes that the client is down and terminates the connection.

TIME-WAIT Timer –

The TIME-WAIT (2MSL) timer is used during connection termination.

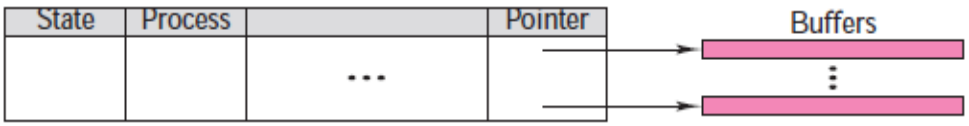
TCP PACKAGE –

The package involves tables called transmission control blocks, a set of timers, and three software modules: a main module, an input processing module, and an output processing module. Figure – shows these five components and their interactions.



Transmission Control Blocks (TCBs) –

TCP is a connection-oriented transport protocol. A connection may be open for a long period of time. To control the connection, TCP uses a structure to hold information about each connection. This is called a *transmission control block* (TCB). Because at any time there can be several connections, TCP keeps an array of TCBs in the form of a table. The table is usually referred to as the TCB (see Figure).



Many fields can be included in each TCB. We mention only the most common ones here.
State – This field defines the state of the connection according to the state transition diagram.
Process – This field defines the process using this connection at this machine as a client or a server.
Local IP address – This field defines the IP address of the local machine used by this connection.
Local port number –This field defines the local port number used by this connection.
Remote IP address – This field defines the IP address of the remote machine used by this connection.
Remote port number – This field defines the remote port number used by this connection.
Interface – This field defines the local interface.
Local window – This field, which can comprise several subfields, holds information about the window at the local TCP.
Remote window –This field, which can comprise several subfields, holds information about the window at the remote TCP.
Sending sequence number – This field holds the sending sequence number.
Receiving sequence number – This field holds the receiving sequence number.
Sending ACK number – This field holds the value of the ACK number sent.
Round-trip time – Several fields may be used to hold information about the RTT.
Time-out values – Several fields can be used to hold the different time-out values such as the retransmission time-out, persistence time-out, keepalive time-out, and so on.
Buffer size – This field defines the size of the buffer at the local TCP.
Buffer pointer –This field is a pointer to the buffer where the received data are kept until they are read by the application.

Main Module –

- The main module is invoked by an arriving TCP segment, a time-out event, or a message from an application program. This is a very complicated module because the action to be taken depends on the current state of the TCP. Several approaches have been used to implement the state transition diagram including using a process for each state, using a table (two-dimensional array), and so on.
- The ESTABLISHED state needs further explanation. When TCP is in this state and data or an acknowledgment segment arrives, another module, the input processing module, is called to handle the situation.
- Also, when TCP is in this state and a "send data" message is issued by an application program, another module, the output processing module, is called to handle the situation.

Input Processing Module –

In our design, the input processing module handles all the details needed to process data or an acknowledgment received when TCP is in the ESTABLISHED state. This module sends an ACK if needed, takes care of the window size announcement, does error checking, and so on.

Output Processing Module –

In our design, the output processing module handles all the details needed to send out data received from application program when TCP is in the ESTABLISHED state. This module handles retransmission time-outs, persistent time-outs, and so on.

UNIT IV

SCTP –

- Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced.
- These new applications, such as IUA (ISDN over IP), M2UA and M3UA (telephony signaling), H.248 (media gateway control), H.323 (IP telephony), and SIP (IP telephony), need a more sophisticated service than TCP can provide. SCTP provides this enhanced performance and reliability.
- SCTP is a *message-oriented, reliable* protocol that combines the best features of UDP and TCP.

SCTP Services –

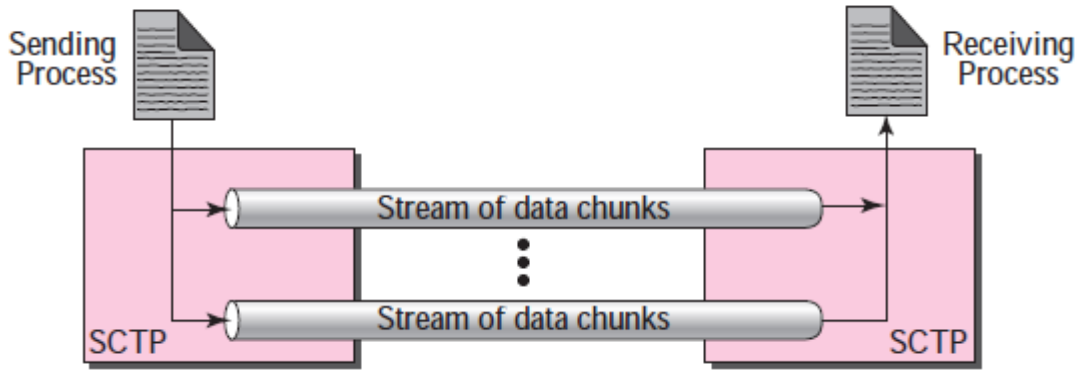
Process-to-Process Communication –

SCTP uses all well-known ports in the TCP space. Table lists some extra port numbers used by SCTP.

Protocol	*Port Number*	*Description*
IUA	9990	ISDN over IP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718, 1719, 1720, 11720	IP telephony
SIP	5060	IP telephony

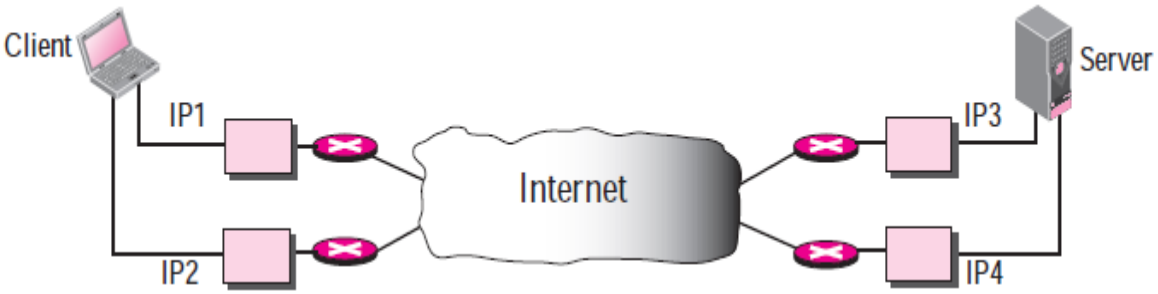
Multiple Streams –

- Each connection between a TCP client and a TCP server involves one single stream. The problem with this approach is that a loss at any point in the stream blocks the delivery of the rest of the data.
- This can be acceptable when we are transferring text; it is not when we are sending real-time data such as audio or video. SCTP allows multistream service in each connection, which is called association in SCTP terminology. If one of the streams is blocked, the other streams can still deliver their data.



Multihoming –

- A TCP connection involves one source and one destination IP address. This means that even if the sender or receiver is a multihomed host (connected to more than one physical address with multiple IP addresses), only one of these IP addresses per end can be utilized during the connection.
- An SCTP association, on the other hand, supports multihoming service. The sending and receiving host can define multiple IP addresses in each end for an association. In this fault-tolerant approach, when one path fails, another interface can be used for data delivery without interruption.
- This fault-tolerant feature is very helpful when we are sending and receiving a real-time payload such as Internet telephony. Figure – shows the idea of multihoming.



- In Figure, the client is connected to two local networks with two IP addresses. The server is also connected to two networks with two IP addresses. The client and the server can make an association, using four different pairs of IP addresses.
- However, note that in the current implementations of SCTP, only one pair of IF addresses can be chosen for normal communication; the alternative is used if the main choice fails. In other words, at present, SCTP does not allow load sharing between different paths.

Full-Duplex Communication –

- Like TCP, SCTP offers full-duplex service, in which data can flow in both directions at the same time. Each SCTP then has a sending and receiving buffer, and packets are sent in both directions.

Connection-Oriented Service –

Like TCP, SCTP is a connection-oriented protocol. However, in SCTP, a connection is called an association. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two SCTPs establish an association between each other.
2. Data are exchanged in both directions.
3. The association is terminated.

Reliable Service –

- SCTP, like TCP, is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

SCTP Features –

Transmission Sequence Number –

- The unit of data in TCP is a byte. Data transfer in TCP is controlled by numbering bytes by using a sequence number. On the other hand, the unit of data in SCTP is a DATA chunk which may or may not have a one-to-one relationship with the message coming from the process because of fragmentation, Data transfer in SCTP is controlled by numbering the data chunks.
- SCTP uses a transmission sequence number (TSN) to number the data chunks. In other words, the TSN in SCTP plays the analogous role to the sequence number in TCP.

Stream Identifier –

- In TCP, there is only one stream in each connection. In SCTP, there may be several streams in each association. Each stream in SCTP needs to be identified by using a stream identifier (SI).
- Each data chunk must carry the SI in its header so that when it arrives at the destination, it can be properly placed in its stream.

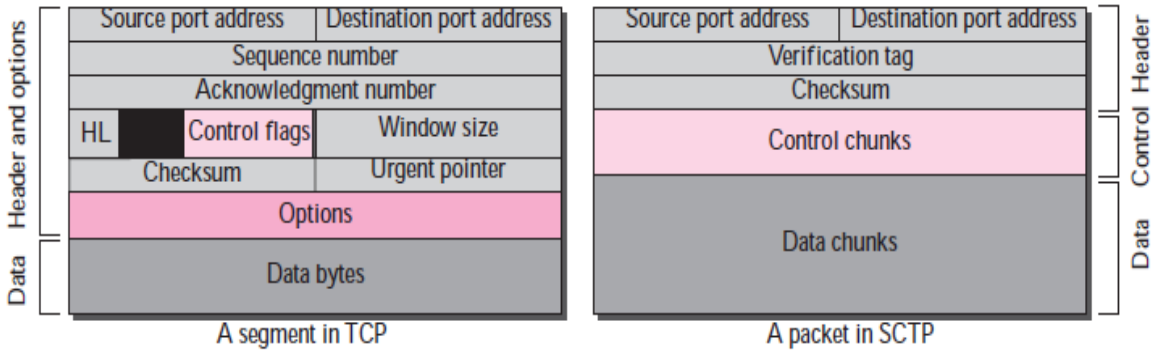
Stream Sequence Number –

- When a data chunk arrives at the destination SCTP, it is delivered to the appropriate stream and in the proper order. This means that, in addition to an SI, SCTP defines each data chunk in each stream with a stream sequence number (SSN).

Packets –

- In TCP, a segment carries data and control information. Data are carried as a collection of bytes; control information is defined by six control flags in the header.
- The design of SCTP is totally different: data are carried as data chunks, control information is carried as control chunks. Several control chunks and data chunks can be packed together in a packet.
- A packet in SCTP plays the same role as a segment in TCP. Figure – compares a segment in TCP and a packet in SCTP.
- List the differences between an SCTP packet and a TCP segment:

1. The control information in TCP is part of the header; the control information in SCTP is included in the control chunks. There are several types of control chunks; each is used for a different purpose.



Comparison between a TCP segment and an SCTP packet

2. The data in a TCP segment treated as one entity; an SCTP packet can carry several data chunks; each can belong to a different stream.
3. The options section, which can be part of a TCP segment, does not exist in an SCTP packet. Options in SCTP are handled by defining new chunk types.
4. The mandatory part of the TCP header is 20 bytes, while the general header in SCTP is only 12 bytes. The SCTP header is shorter due to the following:
   a. An SCTP sequence number (TSN) belongs to each data chunk and hence is located in the chunk's header.
   b. The acknowledgment number and window size are part of each control chunk.
   c. There is no need for a header length field (shown as HL in the TCP segment) because there are no options to make the length of the header variable; the SCTP header length is fixed (12 bytes).
   d. There is no need for an urgent pointer in SCTP.
5. The checksum in TCP is 16 bits; in SCTP, it is 32 bits.
6. The verification tag in SCTP is an association identifier, which does not exist in TCP. In TCP, the combination of IP and port addresses defines a connection; in SCTP we may have multihorning using different IP addresses. A unique verification tag is needed to define each association.
7. TCP includes one sequence number in the header, which defines the number of the first byte in the data section. An SCTP packet can include several different data chunks. TSNs, SIs, and SSNs define each data chunk.
8. Some segments in TCP that carry control information (such as SYN and FIN) need to consume one sequence number; control chunks in SCTP never use a TSN, SI, or SSN. These three identifiers belong only to data chunks, not to the whole packet.

Acknowledgment Number –

- TCP acknowledgment numbers are byte-oriented and refer to the sequence numbers. SCTP acknowledgment numbers are chunk-oriented. They refer to the TSN. A second difference between TCP and SCTP acknowledgments is the control information.

Flow Control –

- Like TCP, SCTP implements flow control to avoid overwhelming the receiver.

Error Control –

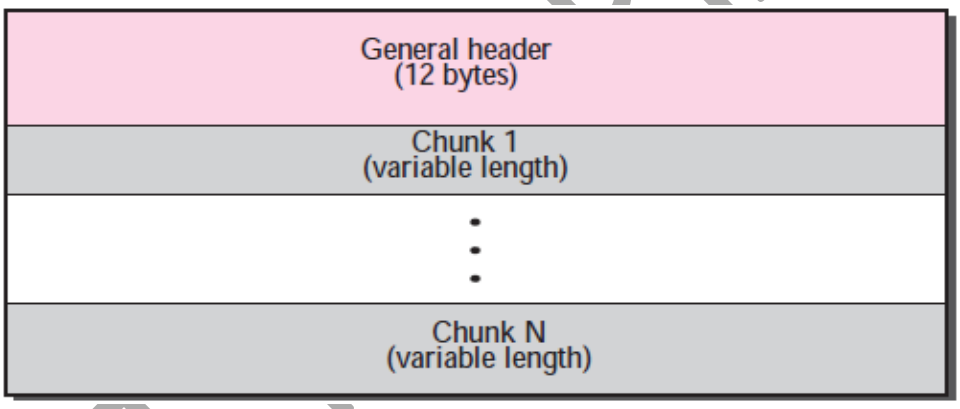
- Like TCP, SCTP implements error control to provide reliability. TSN numbers and acknowledgment numbers are used for error control.

Congestion Control –

- Like TCP, SCTP implements congestion control to determine how many data chunks can be injected into the network.

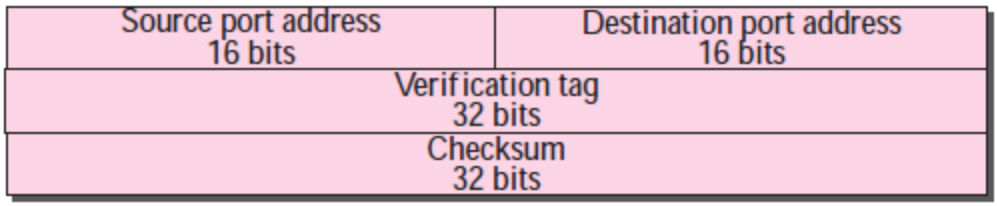
Packet Format –

An SCTP packet has a mandatory general header and a set of blocks called chunks. There are two types of chunks: control chunks and data chunks. A control chunk controls and maintains the association; a data
chunk carries user data. In an SCTP packet, control chunks come before data chunks. Figure – shows the general format of an SCTP packet.



General Header –

The general header (packet header) defines the endpoints of each association to which the packet belongs, guarantees that the packet belongs to a particular association, and preserves the integrity of the contents of the packet including the header itself. The format of the general header is shown in Figure –



There are four fields in the general header:

Source port address –

- This is a 16-bit field that defines the port number of the process sending the packet.

Destination port address –

- This is a 16-bit field that defines the port number of the process receiving the packet.

Verification tag –

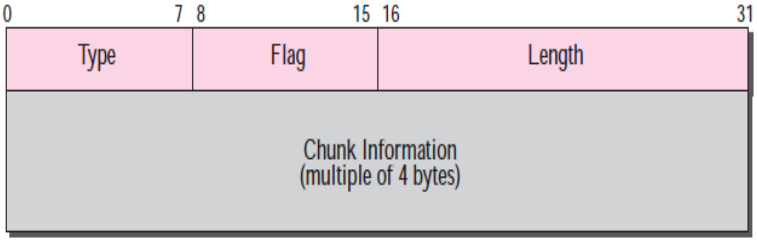
- This is a number that matches a packet to an association. This prevents a packet from a previous association from being mistaken as a packet in this association. It serves as an identifier for the association; it is repeated in every packet during the association. There is a separate verification used for each direction in the association.

Checksum –

- This 32-bit field contains a CRC-32 checksum. Note that the size of the checksum is increased from 16 (in UDP, TCP, and IP) to 32 bits to allow the use of the CRC-32 checksum.

Chunks –

- Control information or user data are carried in chunks. The first three fields are common to all chunks; the information field depends on the type of chunk.
- The important point to remember is that SCTP requires the information section to be a multiple of 4 bytes; if not, padding bytes (eight as) are added at the end of the section.



The description of the common fields are as follows:

Type. This 8-bit field can define up to 256 types of chunks. Only a few have been defined so far; the rest are reserved for future use. See Table 16.2 for a list of chunks and their descriptions.
Flag. This 8-bit field defines special flags that a particular chunk may need. Each bit has a different meaning depending on the type of chunk.
Length. Since the size of the information section is dependent on the type of chunk, we need to define the chunk boundaries. This 16-bit field defines the total size of the chunk, in bytes, including the type, flag, and length fields.

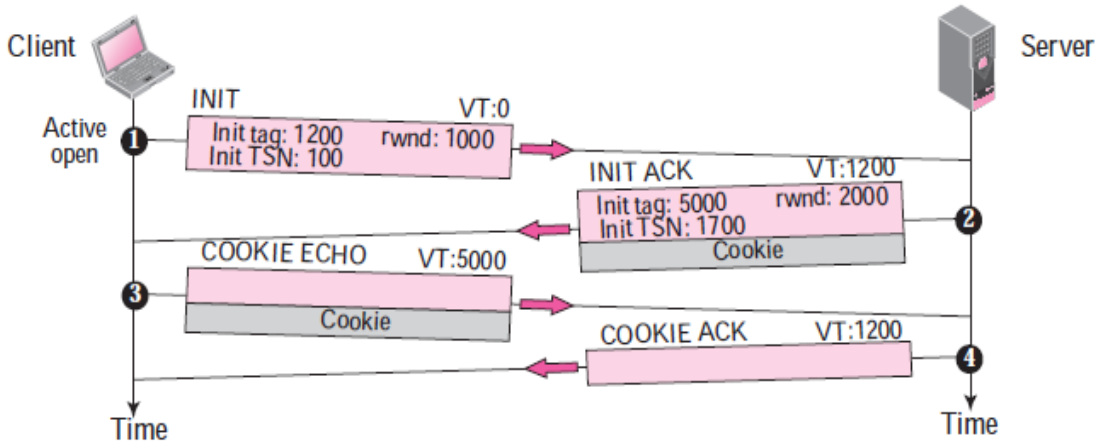
An SCTP Association –

SCTP, like TCP, is a connection-oriented protocol. However, a connection in SCTP is called an *association* to emphasize multihoming. A connection in SCTP is called an association.

Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INITACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Aborts an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIEACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARDTSN	For adjusting cumulative TSN

Association Establishment –

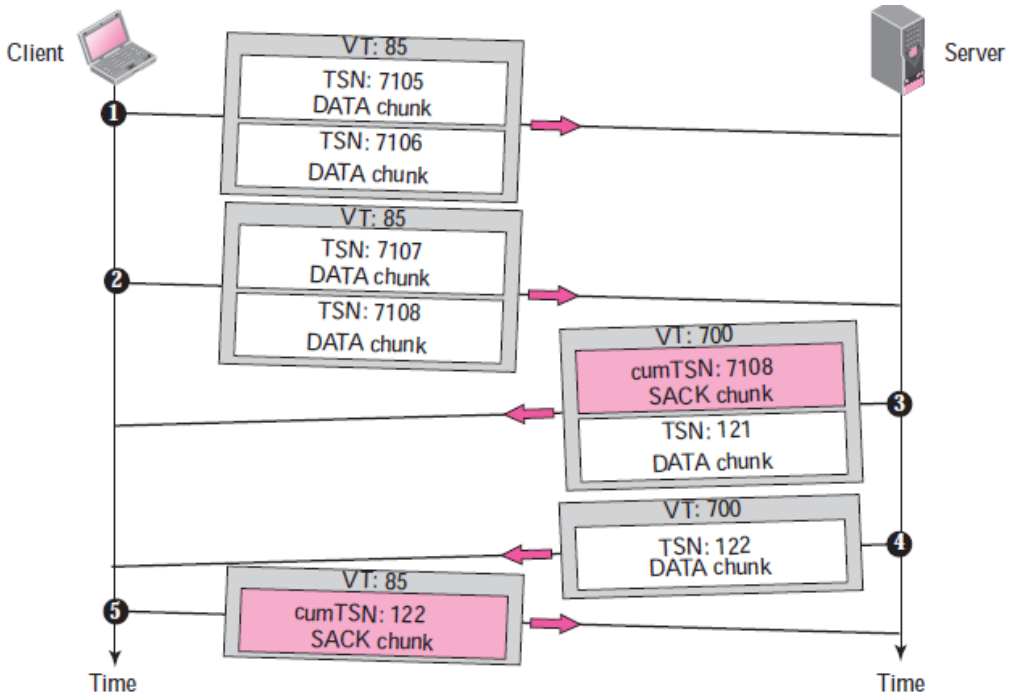
- Association establishment in SCTP requires a four-way handshake. In this procedure, a process, normally a client, wants to establish an association with another process, normally a server, using SCTP as the transport layer protocol.
- Similar to TCP, the SCTP server needs to be prepared to receive any association (passive open). Association establishment, however, is initiated by the client (active open). SCTP association establishment is shown in Figure – The steps, in a normal situation, are as follows:
  1. The client sends the first packet, which contains an INIT chunk.
  2. The server sends the second packet, which contains an INIT ACK chunk.
  3. The client sends the third packet, which includes a COOKIE ECHO chunk. This is a very simple chunk that echoes, without change, the cookie sent by the server. SCTP allows the inclusion of data chunks in this packet.
  4. The server sends the fourth packet, which includes the COOKIE ACK chunk that acknowledges the receipt of the COOKIE ECHO chunk. SCTP allows the inclusion of data chunks with this packet.



No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk. A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.

Data Transfer –

- The whole purpose of an association is to transfer data between two ends. After the association is established, bidirectional data transfer can take place. The client and the server can both send data. Like TCP, SCTP supports piggybacking.
- There is a major difference, however, between data transfer in TCP and SCTP. TCP receives messages from a process as a stream of bytes without recognizing any boundary between them. The process may insert some boundaries for its peer use, but TCP treats that mark as part of the text.
- SCTP, on the other hand, recognizes and maintains boundaries. Each message coming from the process is treated as one unit and inserted into a DATA chunk unless it is fragmented.In this figure a client sends four DATA chunks and receives two DATA chunks from the server.



1.The client sends the first packet carrying two DATA chunks with TSNs 7105 and 7106.
2.The client sends the second packet carrying two DATA chunks with TSNs 7107 and 7108.
3.The third packet is from the server. It contains the SACK chunk needed to acknowledge the receipt of DATA chunks from the client. Contrary to TCP, SCTP acknowledges the last in-order TSN received, not the next expected. The third packet also includes the first DATA chunk from the server with TSN 121.
4.After a while, the server sends another packet carrying the last DATA chunk with TSN 122, but it does not include a SACK chunk in the packet because the last DATA chunk received from the client was already acknowledged.
5.Finally, the client sends a packet that contains a SACK chunk acknowledging the receipt of the last two DATA chunks from the server.

The acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.

Multihoming Data Transfer –

- Multihoming allows both ends to define multiple IP addresses for communication. However, only one of these addresses can be defined as the primary address; the rest are alternative addresses.
- The primary address is defined during association establishment. The interesting point is that the primary address of an end is determined by the other end. In other words, a source defines the primary address for a destination.

Multistream –

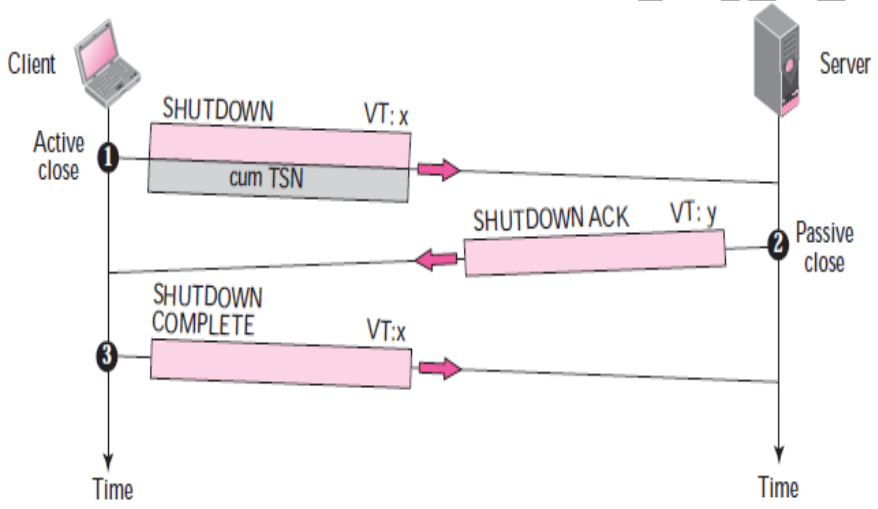
- Delivery One interesting feature of SCTP is the distinction between data transfer and data delivery. SCTP uses TSN numbers to handle data transfer, movement of data chunks between the source and destination.

Fragmentation –

SCTP preserves the boundaries of the message from process to process when creating a DATA chunk from a message if the size of the message (when encapsulated in an IP datagram) does not exceed the MTU of the path.

Association Termination –

- In SCTP, like TCP, either of the two parties involved in exchanging data (client or server) can close the connection. However, unlike TCP, SCTP does not allow a halfclose situation. If one end closes the association, the other end must stop sending new data.
- If any data are left over in the queue of the recipient of the termination request, they are sent and the association is closed. Association **termination** uses three packets, as shown in Figure.



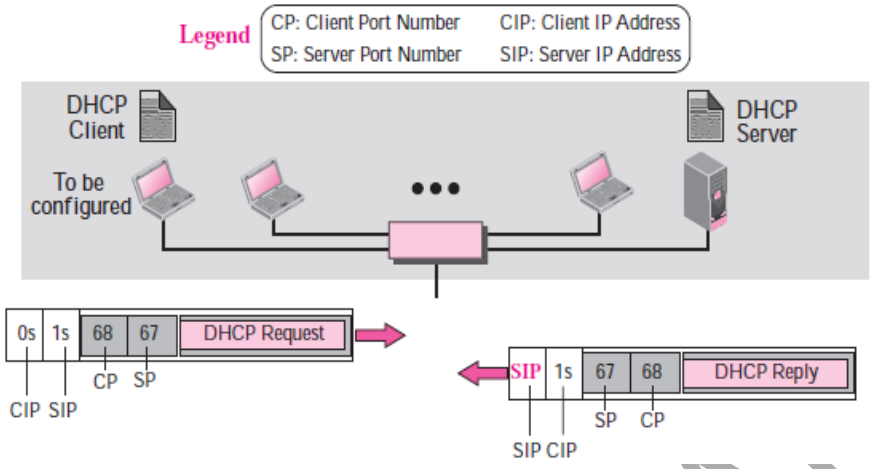
DHCP

The **Dynamic Host Configuration Protocol (DHCP)** is a client/server protocol designed to provide the four pieces of information for a diskless computer or a computer that is booted for the first time.

DHCP OPERATION –

Same Network –

Although the practice is not very common, the administrator may put the client and the server on the same network as shown in Figure –



In this case, the operation as follows:

1. The DHCP server issues a passive open command on UDP port number 67 and waits for a client.
2. A booted client issues an active open command on port number 68. The message is encapsulated in a UDP user datagram, using the destination port number 67 and the source port number 68. The UDP user datagram, in turn, is encapsulated in an IP datagram. The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client uses all 0s as the source address and all 1s as the destination address.
3. The server responds with either a broadcast or a unicast message using UDP source port number 67 and destination port number 68. The response can be unicast because the server knows the IP address of the client. It also knows the physical address of the client, which means it does not need the services of ARP for logical to physical address mapping. However, some systems do not allow the bypassing of ARP, resulting in the use of the broadcast address.

Different Networks –

- As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks. Figure – shows the situation. However, there is one problem that must be solved. The DHCP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. A router receiving such a packet discards it.
- To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this case is called a **relay agent.**
- The relay agent knows the unicast address of a DHCP server and listens for broadcast messages on port 67. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the DHCP server.
- The packet, carrying a unicast destination address, is routed by any router and reaches the DHCP server. The DHCP server knows the message comes from a relay agent because

one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the DHCP client.

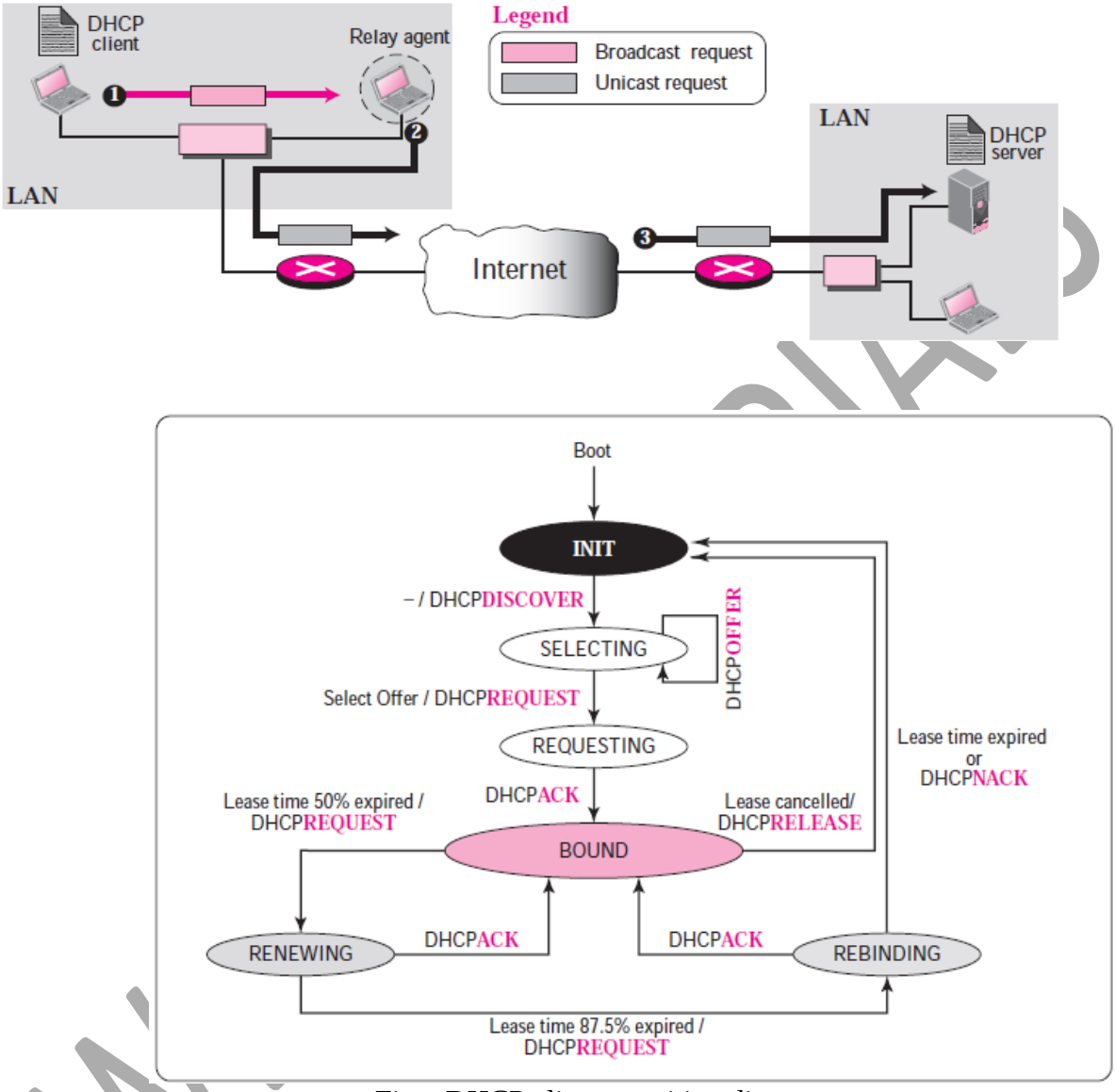


Fig – DHCP client transition diagram

INIT State

- When the DHCP client first starts, it is in the INIT state (initializing state). The client broadcasts a DHCPDISCOVER message (a request message with the DHCPDISCOVER option), using port 67.

SELECTING State

- After sending the DHCPDISCOVER message, the client goes to the selecting state. Those servers that can provide this type of service respond with a DHCPOFFER

message. In these messages, the servers offer an IP address. They can also offer the lease duration. The default is 1 hour. The server that sends a DHCPOFFER locks the offered IP address so that it is not available to any other clients.

- The client chooses one of the offers and sends a DHCPREQUEST message to the selected server. It then goes to the requesting state. However, if the client receives no DHCPOFFER message, it tries four more times, each with a span of 2 seconds. If there is no reply to any of these DHCPDISCOVERs, the client sleeps for 5 minutes before trying again.

REQUESTING State

- The client remains in the requesting state until it receives a DHCPACK message from the server that creates the binding between the client physical address and its IP address. After receipt of the DHCPACK, the client goes to the bound state.

BOUND State

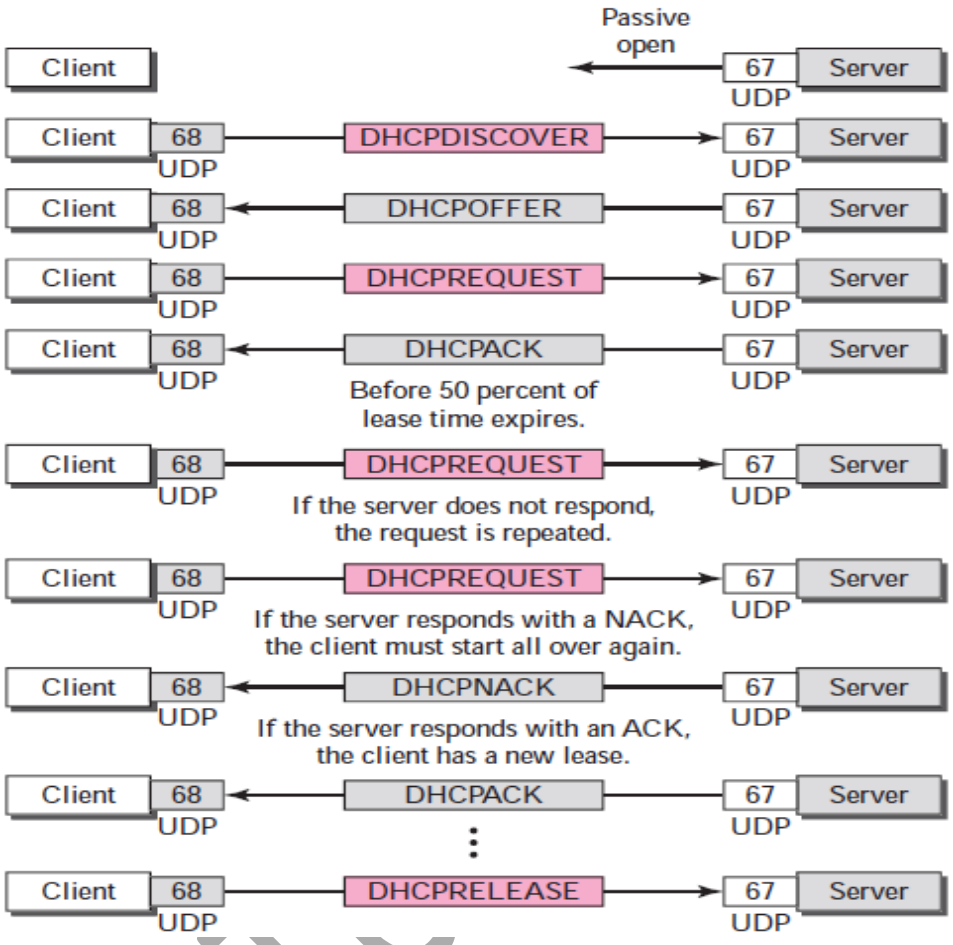
- In this state, the client can use the IP address until the lease expires. When 50 percent of the lease period is reached, the client sends another DHCPREQUEST to ask for renewal. It then goes to the renewing state. When in the bound state, the client can also cancel the lease and go to the initializing state.

RENEWING State

- The client remains in the renewing state until one of two events happens. It can receive a DHCPACK, which renews the lease agreement. In this case, the client resets its timer and goes back to the bound state. Or, if a DHCPACK is not received, and 87.5 percent of the lease time expires, the client goes to the rebinding state.

REBINDING State

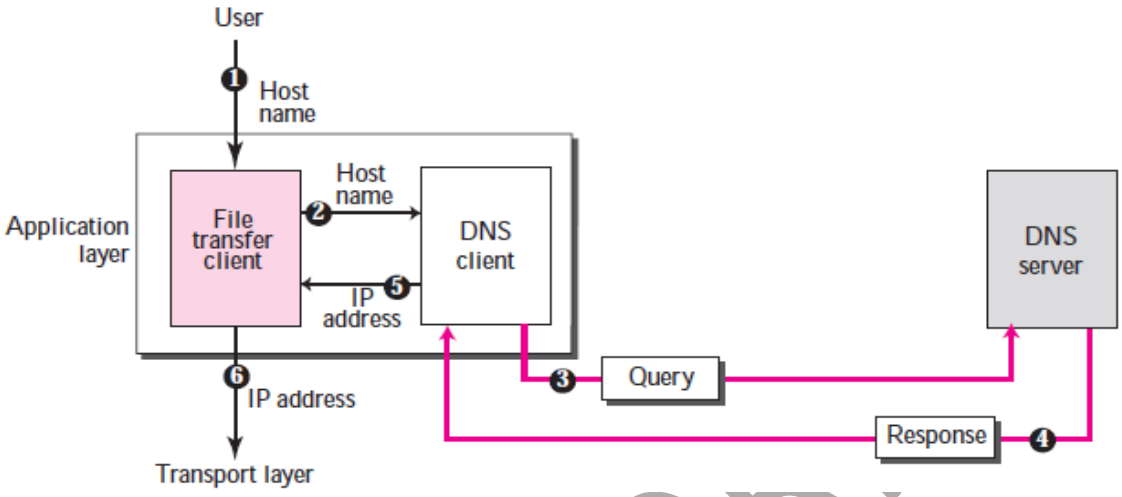
- The client remains in the rebinding state until one of three events happens. If the client receives a DHCPNACK or the lease expires, it goes back to the initializing state and tries to get another IP address. If the client receives a DHCPACK, it goes to the bound state and resets the timer.



NEED FOR DNS –

- To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.
- When the Internet was small, mapping was done using a *host file*. The host file had only two columns: name and address. Every host could store the host file on its disk and update it periodically from a master host file. When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping.
- Today, however, it is impossible to have one single host file to relate every address with a name and vice versa. The host file would be too large to store in every host. In addition, it would be impossible to update all the host files every time there is a change.
- One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet.
- Another solution, the one used today, is to divide this huge amount of information into smaller parts and store each part on a different computer. In this method, the host that needs mapping can contact the closest computer holding the needed information.

- This method is used by the **Domain Name System (DNS).**Figure shows how TCP/IP uses a DNS client and a DNS server to map a name to an address; the reverse mapping is similar.



In Figure, a user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as *forouzan.com.* However, the TCP/IP suite needs the IP address of the file transfer server to make the connection. The following six steps map the host name to an IP address.

1. The user passes the host name to the file transfer client.

2. The file transfer client passes the host name to the DNS client.

3. We know from Chapter 18 that each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.

4. The DNS server responds with the IP address of the desired file transfer server.

5. The DNS client passes the IP address to the file transfer server.

6. The file transfer client now uses the received IP address to access the file transfer server.

NAME SPACE –

- The names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique.
- A **name space** that maps each address to a unique name can be organized in two ways: flat or hierarchical.

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; if they do, it has no meaning.

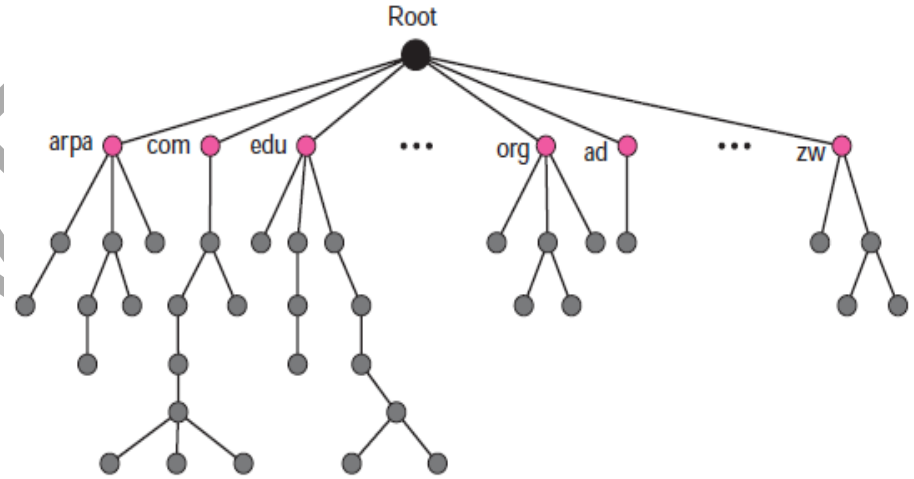
- The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical 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. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization.
- The responsibility of the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources.
- For example, assume two colleges and a company call one of their computers *challenger*. The first college is given a name by the central authority such as *fhda.edu*, the second college is given the name *berkeley.edu*, and the company is given the name *smart.com.*
- When each of these organizations adds the name *challenger* to the name they have already been given, the end result is three distinguishable names: *challenger.fhda.edu, challenger.berkeley.edu,* and *challenger.smart.com*. The names are unique without the need for assignment by a central authority. The central authority controls only part of the name, not the whole.

Domain Name Space –

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

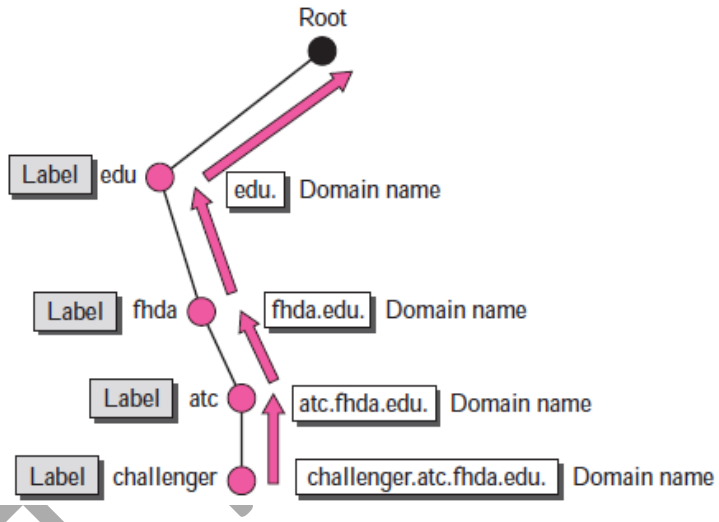


Label –

Each node in the tree has a **label,** which is a string with a maximum of 63 characters. DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name –

Each node in the tree has a domain name. A full **domain name** is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Figure – shows some domain names.



Fully Qualified Domain Name (FQDN) –

- If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN).** An FQDN is a domain name that contains the full name of a host.
- It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name is the FQDN of a computer named *challenger* installed at the Advanced Technology Center (ATC) at De Anza College.
- A DNS server can only match an FQDN to an address. Note that the name must end with a null label, but because null means nothing, the label ends with a dot (.).

challenger.atc.fhda.edu.

Partially Qualified Domain Name (PQDN)

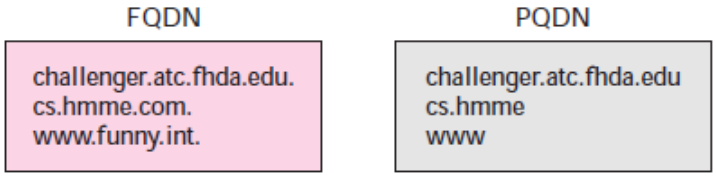
- If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN).** A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client.
- Here the resolver can supply the missing part, called the *suffix*, to create an FQDN. For example, if a user at the *fhda.edu.* site wants to get the IP address of the challenger computer, he or she can define the partial name.

challenger

The DNS client adds the suffix *atc.fhda.edu*. before passing the address to the DNS server. The DNS client normally holds a list of suffixes. The following can be the list of suffixes at De Anza College. The null suffix defines nothing. This suffix is added when the user defines an FQDN.

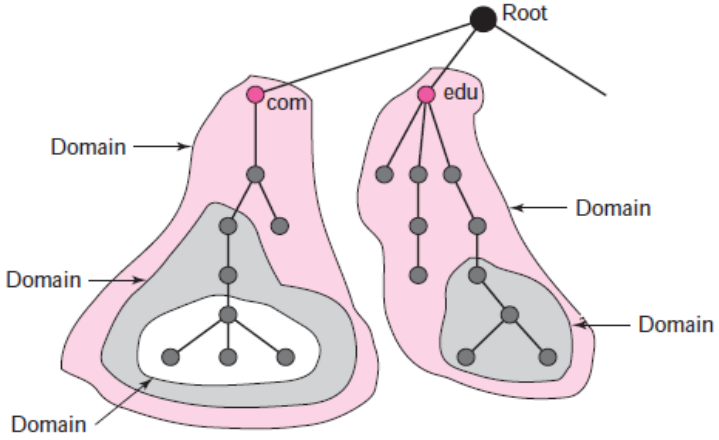
atc.fhda.edu fhda.edu null

Figure – shows some FQDNs and PQDNs.



Domain

A **domain** is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree. Figure – shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).



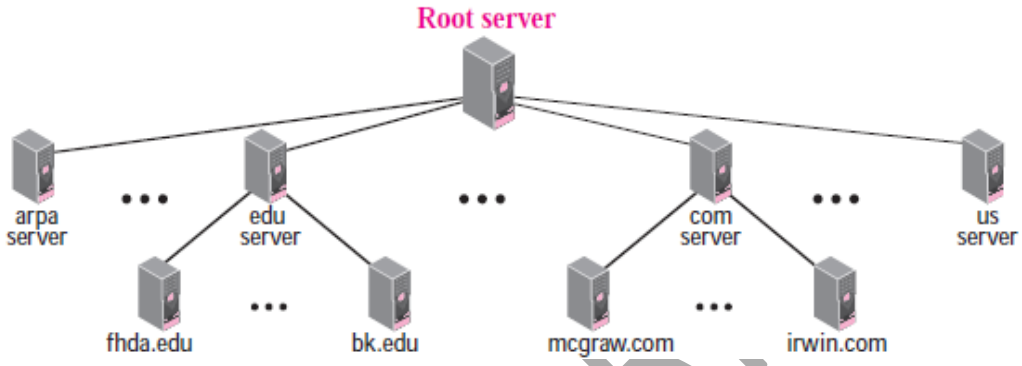
Distribution of Name Space

- The information contained in the domain name space must be stored. However, it is very inefficient and also not reliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not reliable because any failure makes the data inaccessible.

Hierarchy of Name Servers

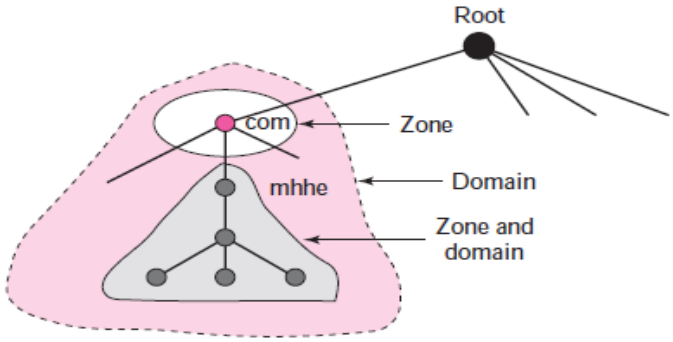
- The solution to these problems is to distribute the information among many computers called **DNS servers.** One way to do this is to divide the whole space into many domains based on the first level.

- In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains).
- Each server can be responsible (authoritative) for either a large or small domain. In other words, we have a hierarchy of servers in the same way that we have a hierarchy of names (see Figure).



Zone

- Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a **zone.**



- We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the "domain" and the "zone" refer to the same thing.
- The server makes a database called a *zone file* and keeps all the information for every node under that domain. However, if a server divides its domain into subdomains and delegates part of its authority to other servers, "domain" and "zone" refer to different things.

Root Server

A **root server** is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers.

Primary and Secondary Servers

- DNS defines two types of servers: primary and secondary. A **primary server** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.
- A **secondary server** is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk.
- The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.
- A primary server loads all information from the disk file; the secondary server loads all information from the primary server. When the secondary downloads information from the primary, it is called zone transfer.

DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain (see Figure).

Generic Domains

The **generic domains** define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database(see Figure).

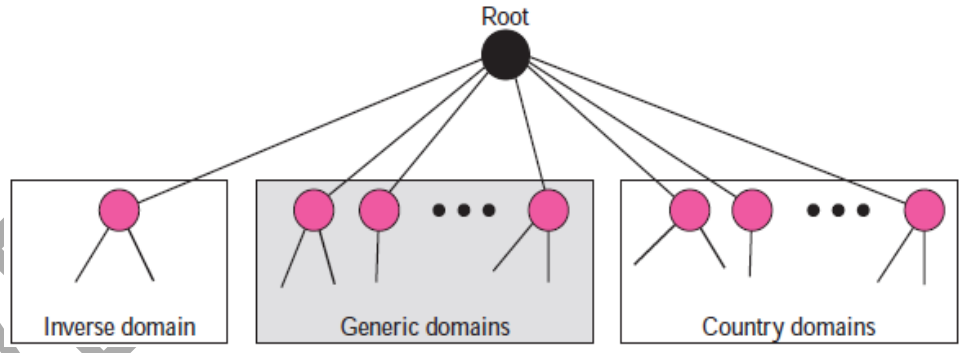
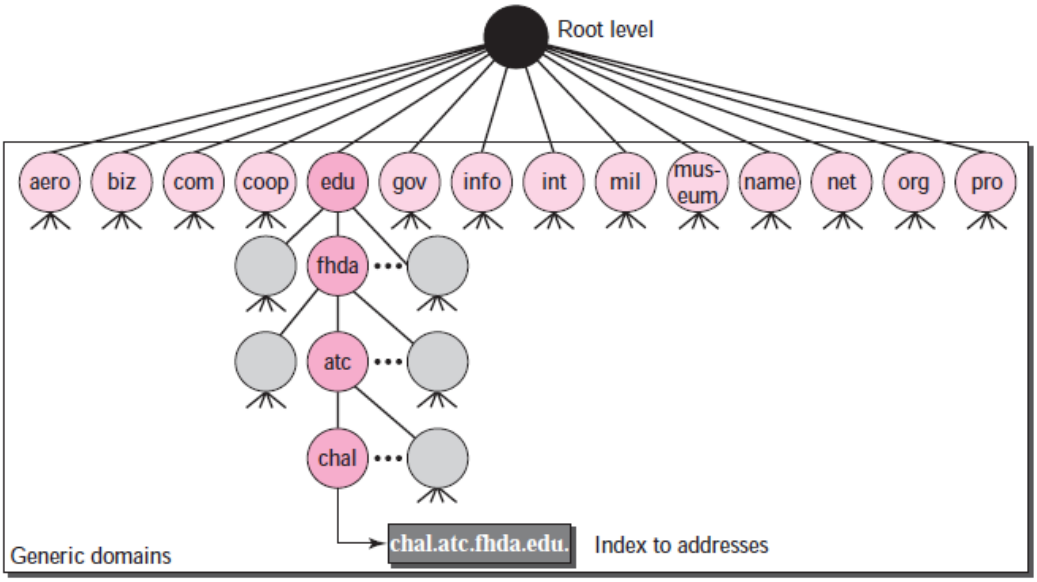


***Figure** - DNS used in the Internet*

Figure – *Generic domains*



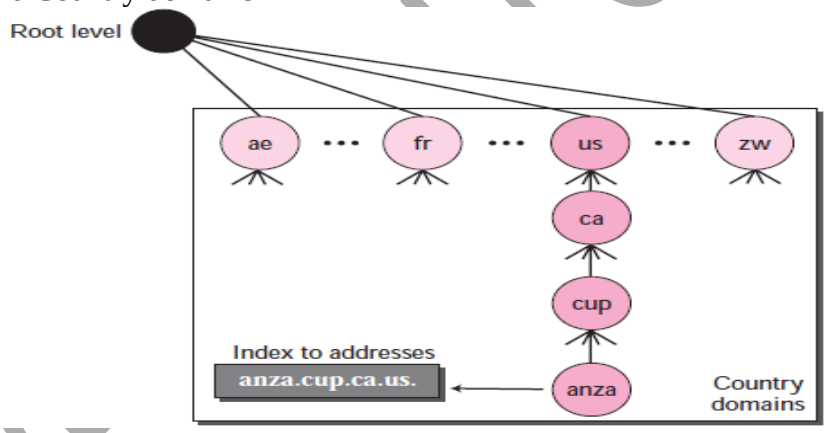
Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table.

Country Domains

The **country domains** section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).
Figure – shows the country domains section. The address *anza.cup.ca.us* can be translated to De Anza College in Cupertino in California in the United States.

Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other non-profit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

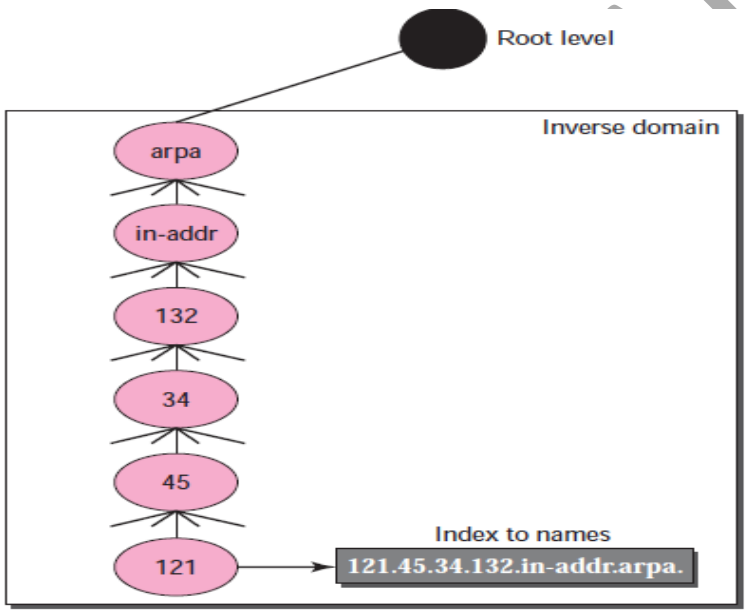
Figure 19.10 Country domains



Inverse Domain

- The **inverse domain** is used to map an address to a name. This may happen, for example, when a server has received a request from a client to do a task. Although the server has a file that contains a list of authorized clients, only the IP address of the client (extracted from the received IP packet) is listed.
- The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list.
- This type of query is called an inverse or pointer (PTR) query. To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called *arpa* (for historical reasons).

- The second level is also one single node named *in-addr* (for inverse address). The rest of the domain defines IP addresses.The servers that handle the inverse domain are also hierarchical.
- This means the netid part of the address should be at a higher level than the subnetid part, and the subnetid part higher than the hostid part. In this way, a server serving the whole site is at a higher level than the servers serving each subnet. This configuration makes the domain look inverted when compared to a generic or country domain.
- To follow the convention of reading the domain labels from the bottom to the top, an IP address such as 132.34.45.121 (a class B address with netid 132.34) is read as 121.45.34.132.in-addr. arpa. See Figure – for an illustration of the inverse domain configuration.



Registrar –

How are the new domains added to DNS? This is done through a **registrar,** a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged.

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.**

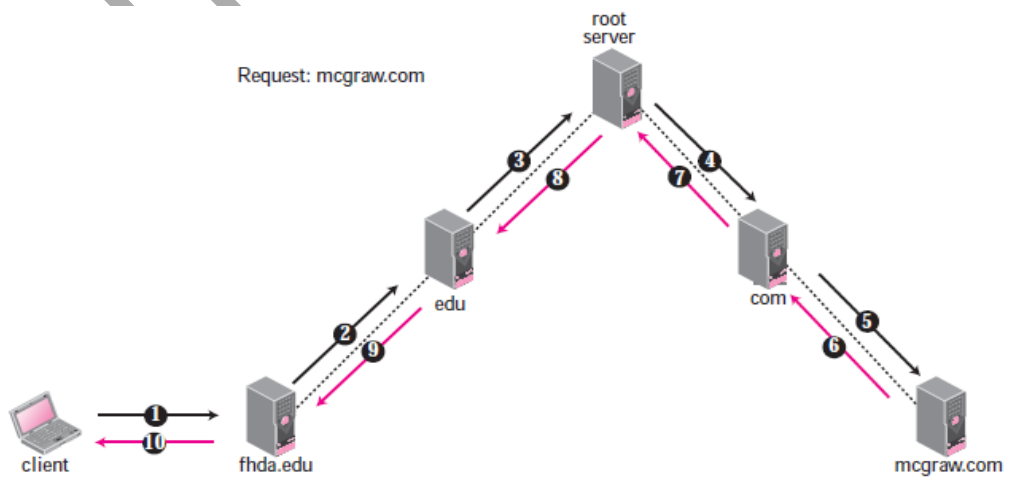
- The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.
- After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

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 two labels, *in-addr* and *arpa*, are appended to create a domain acceptable by the inverse domain section.
- For example, if the resolver receives the IP address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is "*121.45.34.132.in-addr.arpa*.", which is received by the local DNS and resolved.

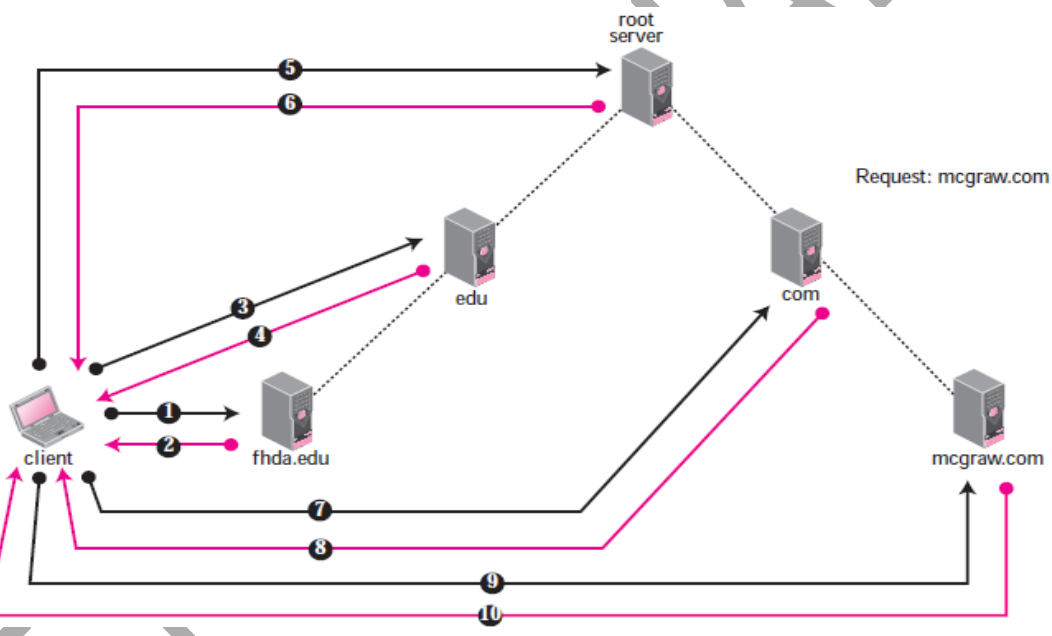
Recursive Resolution –

- Figure – shows the 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.



Iterative Resolution –

- If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query.
- The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Now the client must repeat the query to the third server.
- This process is called *iterative* because the client repeats the same query to multiple servers. In Figure – the client queries five servers before it gets an answer from the mcgraw.com server.



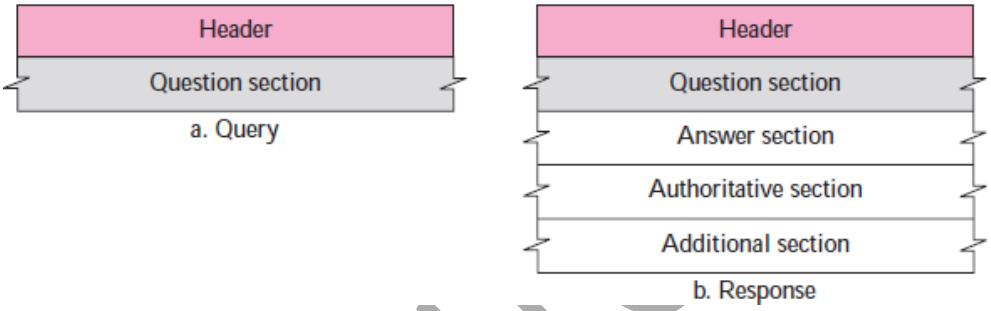
Caching –

- Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called **caching.**
- When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and resolve the problem.
- However, to inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as *unauthoritative.*
- If a server caches a mapping for a long time, it may send an outdated mapping to the client. To counter this, two techniques are used. First, the authoritative server always adds information to the mapping called *time-to-live* (TTL).

- It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server.
- Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically and those mappings with an expired TTL must be purged.

DNS MESSAGES –

DNS has two types of messages: query and response. Both types have the same format. The query message consists of a header and question records; the response message consists of a header, question records, answer records, authoritative records, and additional records (see Figure).



Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is 12 bytes and its format is shown in Figure.

Identification	Flags
Number of question records	Number of answer records (All 0s in query message)
Number of authoritative records (All 0s in query message)	Number of additional records (All 0s in query message)

The header fields are as follows:

Identification – This is a 16-bit field used by the client to match the response with the query. The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response.

Flags – This is a 16-bit field consisting of the subfields shown in Figure.

A brief description of each flag subfield follows.

QR (query/response). This is a 1-bit subfield that defines the type of message. If it is 0, the message is a query. If it is 1, the message is a response.

OpCode. This is a 4-bit subfield that defines the type of query or response (0 if standard, 1 if inverse, and 2 if a server status request).

AA (authoritative answer). This is a 1-bit subfield. When it is set (value of 1) it means that the name server is an authoritative server. It is used only in a response message.

TC (truncated). This is a 1-bit subfield. When it is set (value of 1), it means that the response was more than 512 bytes and truncated to 512. It is used when DNS uses the services of UDP (see Section 19.8 on Encapsulation).

RD (recursion desired). This is a 1-bit subfield. When it is set (value of 1) it means the client desires a recursive answer. It is set in the query message and repeated in the response message.

RA (recursion available). This is a 1-bit subfield. When it is set in the response, it means that a recursive response is available. It is set only in the response message.

Reserved. This is a 3-bit subfield set to 000.

rCode. This is a 4-bit field that shows the status of the error in the response. Of course, only an authoritative server can make such a judgment. Table – shows the possible values for this field.

Value	*Meaning*	*Value*	*Meaning*
0	No error	4	Query type not supported
1	Format error	5	Administratively prohibited
2	Problem at name server	6–15	Reserved
3	Domain reference problem		

Number of question records – This is a 16-bit field containing the number of queries in the question section of the message.

Number of answer records – This is a 16-bit field containing the number of answer records in the answer section of the response message. Its value is zero in the query message.

Number of authoritative records – This is a 16-bit field containing the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message.

Number of additional records – This is a 16-bit field containing the number of additional records in the additional section of a response message. Its value is zero in the query message.

Question Section –

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

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. This section gives information (domain name) about one or more authoritative servers for the query.

Additional Information Section –

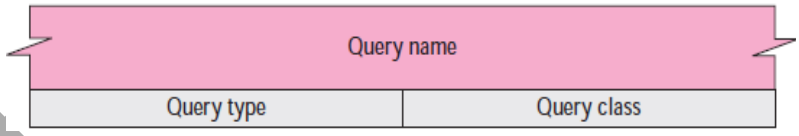
This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional information that may help the resolver. For example, a server may give the domain name of an authoritative server to the resolver in the authoritative section, and include the IP address of the same authoritative server in the additional information section.

TYPES OF RECORDS –

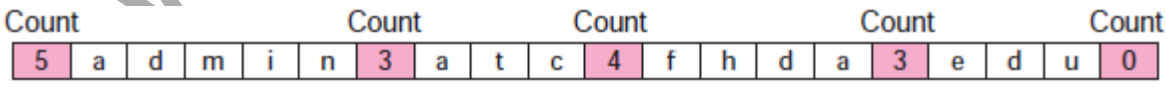
Two types of records are used in DNS. The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

Question Record –

A **question record** is used by the client to get information from a server. This contains the domain name. Figure – shows the format of a question record. The list below describes question record fields.



Query name – This is a variable-length field containing a domain name (see Figure). The count field refers to the number of characters in each section.



Query type. This is a 16-bit field defining the type of query. Table – shows some of the types commonly used. The last two can only be used in a query.

Type	*Mnemonic*	*Description*
1	A	**Address**. A 32-bit IPv4 address. It converts a domain name to an address.
2	NS	**Name server**. It identifies the authoritative servers for a zone.
5	CNAME	**Canonical name**. It defines an alias for the official name of a host.
6	SOA	**Start of authority**. It marks the beginning of a zone.
11	WKS	**Well-known services**. It defines the network services that a host provides.
12	PTR	**Pointer**. It is used to convert an IP address to a domain name.
13	HINFO	**Host information**. It defines the hardware and operating system.
15	MX	**Mail exchange**. It redirects mail to a mail server.
28	AAAA	**Address**. An IPv6 address (see Chapter 26).
252	AXFR	A request for the transfer of the entire zone.
255	ANY	A request for all records.

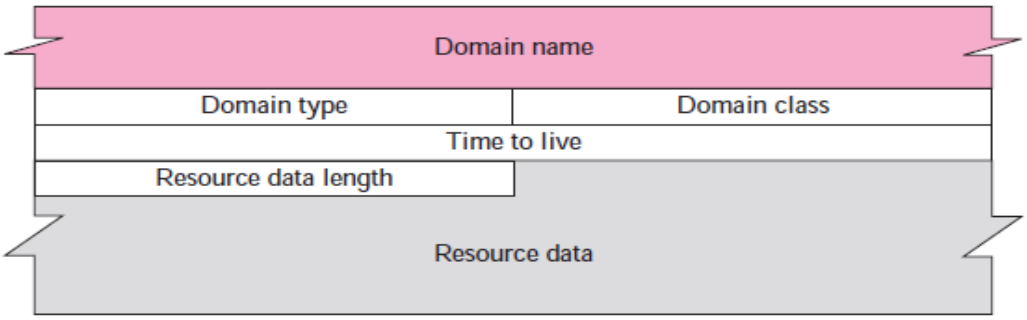
Query class. This is a 16-bit field defining the specific protocol using DNS. Table – shows the current values. In this text we are interested only in class 1(the Internet).

Class	*Mnemonic*	*Description*
1	IN	Internet
2	CSNET	CSNET network (obsolete)
3	CS	The COAS network
4	HS	The Hesiod server developed by MIT

Table - Classes

Resource Record

Each domain name (each node on the tree) is associated with a record called the **resource record.** The server database consists of resource records. Resource records are also what is returned by the server to the client. Figure – shows the format of a resource record.



Domain name. This is a variable-length field containing the domain name. It is a duplication of the domain name in the question record. Since DNS requires the use of compression everywhere a name is repeated, this field is a pointer offset to the corresponding domain name field in the question record.

Domain type. This field is the same as the query type field in the question record except the last two types are not allowed. Refer to Table – Classes for more information.

Domain class. This field is the same as the query class field in the question record (see Table – Classes).

Time-to-live. This is a 32-bit field that defines the number of seconds the answer is valid. The receiver can cache the answer for this period of time. A zero value means that the resource record is used only in a single transaction and is not cached.

Resource data length. This is a 16-bit field defining the length of the resource data.

Resource data. This is a variable-length field containing the answer to the query (in the answer section) or the domain name of the authoritative server (in the authoritative section) or additional information (in the additional information section). The format and contents of this field depend on the value of the type field. It can be one of the following:

A number. This is written in octets. For example, an IPv4 address is a 4-octet integer and an IPv6 address is a 16-octet integer.

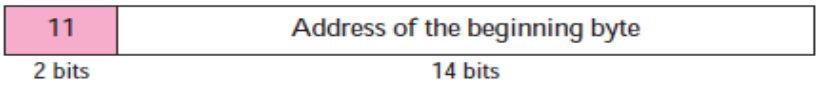
A domain name. Domain names are expressed as a sequence of labels. Each label is preceded by a 1-byte length field that defines the number of characters in the label. Since every domain name ends with the null label, the last byte of every domain name is the length field with the value of 0. To distinguish between a length field and an offset pointer (as we will discuss later), the two high-order bits of a length field are always zero (00). This will not create a problem because the length of a label cannot be more than 63, which is a maximum of 6 bits (111111).

An offset pointer. Domain names can be replaced with an offset pointer. An offset pointer is a 2-byte field with each of the 2 high-order bits set to 1 (11).

A character string. A character string is represented by a 1-byte length field followed by the number of characters defined in the length field. The length field is not restricted like the domain name length field. The character string can be as long as 255 characters (including the length field).

COMPRESSION –

- DNS requires that a domain name be replaced by an offset pointer if it is repeated. For example, in a resource record the domain name is usually a repetition of the domain name in the question record.
- For efficiency, DNS defines a 2-byte offset pointer that points to a previous occurrence of the domain or part of it. The format of the field is shown in Figure.



- The first 2 high-order bits are two 1s to distinguish an offset pointer from a length field. The other 14 bits represent a number that points to the corresponding byte number in the message. The bytes in a message are counted from the beginning of the message with the first byte counted as byte 0.
- For example, if an offset pointer refers to byte 12 (the 13th byte) of the message, the value should be 1100000000001100. Here the 2 leftmost bits define the field as an offset pointer and the other bits define the decimal number 12.

ENCAPSULATION –

- DNS can use either UDP or TCP. In both cases the well-known port used by the server is port 53. UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit. If the size of the response message is more than 512 bytes, a TCP connection is used. In that case, one of two scenarios can occur:
- If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection. For example, if a secondary name server (acting as a client) needs a zone transfer from a primary server, it uses the TCP connection because the size of the information being transferred usually exceeds 512 bytes.
- If the resolver does not know the size of the response message, it can use the UDP port. However, if the size of the response message is more than 512 bytes, the server truncates the message and turns on the TC bit (See Figure). The resolver now opens a TCP connection and repeats the request to get a full response from the server.

REGISTRARS –

How are new domains added to DNS? This is done through a **registrar,** a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged. Today, there are many registrars; their names and addresses can be found at

http://www.intenic.net

To register, the organization needs to give the name of its server and the IP address of the server. For example, a new commercial organization named *wonderful* with a server named *ws* and IP address 200.200.200.5 needs to give the following information to one of the registrars:

Domain name: ws.wonderful.com
IP address: 200.200.200.5

DDNS

- When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file.
- These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.
- The DNS master file must be updated dynamically. The **Dynamic Domain Name System (DDNS)** therefore was devised to respond to this need. In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP to a primary DNS server.
- The primary server updates the zone. The secondary servers are notified either actively or passively. In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes. In either case, after being notified about the change, the secondary requests information about the entire zone (zone transfer).

- To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

SECURITY OF DNS –

DNS is one of the most important systems in the Internet infrastructure; it provides crucial services to the Internet users. Applications such as Web access or e-mail are heavily dependent on the proper operation of DNS. DNS can be attacked in several ways including:

1. The attacker may read the response of a DNS server to find the nature or names of sites the user mostly accesses. This type of information can be used to find the user's profile. To prevent this attack, DNS message needs to be confidential.

2. The attacker may intercept the response of a DNS server and change it or create a totally new bogus response to direct the user to the site or domain the attacker wishes the user to access. This type of attack can be protected using message origin authentication and message integrity.

3. The attacker may flood the DNS server to overwhelm it or eventually crash it. This type of attack can be protected using the provision against denial-of-service attack.

- To protect DNS, IETF has devised a technology named **DNS Security** (**DNSSEC**) that provides the message origin authentication and message integrity using a security service called *digital signature*.
- DNSSEC however, does not provide confidentiality for the DNS messages. There is no specific protection against the denial-of-service attack in the specification of DNSSEC. However, the caching system protects the upper-level servers against this attack to some extent.

UNIT V

TELNET

TELNET is an abbreviation for *TErminaL NETwork*. It is the standard TCP/IP protocol for virtual terminal service as proposed by ISO. TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.
TELNET is a general-purpose client-server application program. TELNET is related to several concepts that

Time-Sharing Environment –

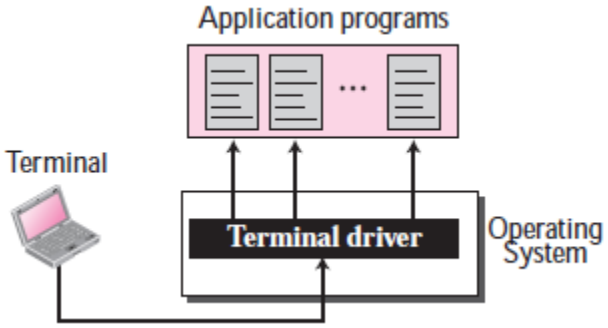
- TELNET was designed at a time when most operating systems, such as UNIX, were operating in a **time-sharing** environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse.
- Even a microcomputer can simulate a terminal with a terminal emulator.
- In a time-sharing environment, all of the processing must be done by the central computer. When a user types a character on the keyboard, the character is usually sent to the computer and echoed to the monitor.
- Time-sharing creates an environment in which each user has the illusion of a dedicated computer. The user can run a program, access the system resources, switch from one program to another, and so on.

Login –

- In a time-sharing environment, users are part of the system with some right to access resources. Each authorized user has an identification and probably a password. The user identification defines the user as part of the system. To access the system, the user logs into the system with a user id or login name.
- The system also includes password checking to prevent an unauthorized user from accessing the resources.

Local Login –

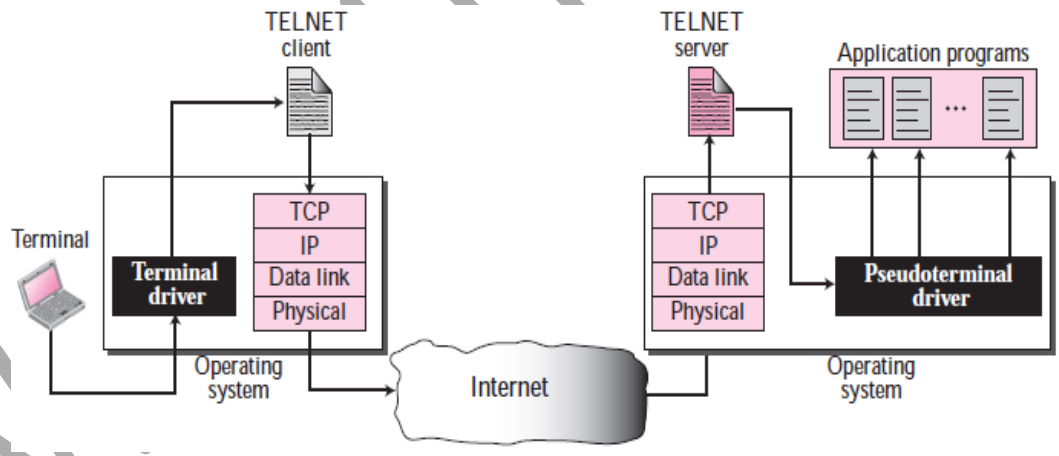
- When a user logs into a local time-sharing system, it is called **local login.** As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system.
- The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility (see Figure). The mechanism, however, is not as simple as it seems because the operating system may assign special meanings to special characters.



- For example, in UNIX some combinations of characters have special meanings, such as the combination of the control character with the character z, which means suspend; the combination of the control character with the character c, which means abort; and so on.

Remote Login –

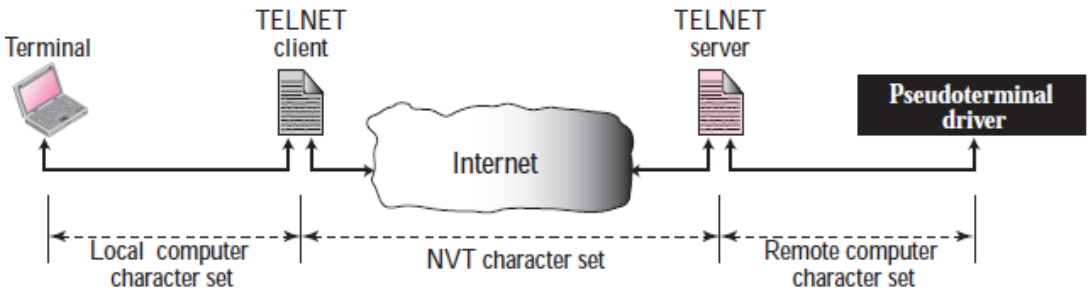
- When a user wants to access an application program or utility located on a remote machine, he or she performs **remote login.** Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters to a universal character set called *Network Virtual Terminal* (NVT) *characters* and delivers them to the local TCP/IP stack (see Figure).



- The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.
- However, the characters cannot be passed directly to the operating system because the remote operating system. is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver.
- The solution is to add a piece of software called a *pseudoterminal driver*, which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

Network Virtual Terminal (NVT) –

- The mechanism to access a remote computer is complex. This is because every computer and its operating system accepts a special combination of characters as tokens.
- For example, the end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d. We are dealing with heterogeneous systems.
- If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the **Network Virtual Terminal (NVT)** character set.
- Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network.
- The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer. For an illustration of this concept in figure –



NVT Character Set –

NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes.

Data Characters –

For data, NVT normally uses what is called NVT ASCII. This is an 8-bit character set in which the seven lowest order bits are the same as US ASCII and the highest order bit is 0 (see Figure). Although it is possible to send an 8-bit ASCII (with the highest order bit set to be 0 or 1), this must first be agreed upon between the client and the server using option negotiation.

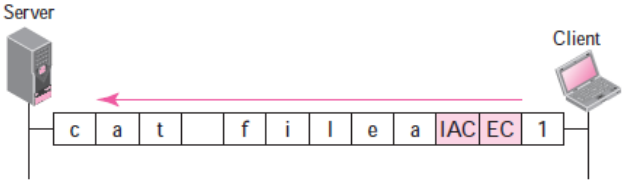
Control Characters –

To send **control characters** between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest order bit is set to 1 (see Figure). Table – lists some of the control characters and their meanings.

Character	*Decimal*	*Binary*	*Meaning*
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
IAC	255	11111111	Interpret (the next character) as control

Embedding –

- TELNET uses only one TCP connection. The server uses the well-known port 23 and the client uses an ephemeral port. The same connection is used for sending both data and control characters. TELNET accomplishes this by embedding the control characters in the data stream.
- However, to distinguish data from control characters, each sequence of control characters is preceded by a special control character called *interpret as control* (IAC).
- For example, imagine a user wants a server to display a file (*file1*) on a remote server. You type – **cat file1** in which *cat* is a Unix command that displays the content of the file on the screen.
- However, the name of the file has been mistyped (*filea* instead of *file1*). The user uses the backspace key to correct this situation. **cat filea<backspace>1**
- However, in the default implementation of TELNET, the user cannot edit locally; the editing is done at the remote server. The backspace character is translated into two remote characters (IAC EC), which is embedded in the data and sent to the remote server. What is sent to the server is shown in Figure –



Options –

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal.
Users with simpler terminals can use default features. Table – shows some common options.

Code	*Option*	*Meaning*
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go-ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

The option descriptions are as follows:

Binary. This option allows the receiver to interpret every 8-bit character received, except IAC, as binary data. When IAC is received, the next character or characters are interpreted as commands. However, if two consecutive IAC characters are received, the first is discarded and the second is interpreted as data.

Echo. This option allows the server to echo data received from the client. This means that every character sent by the client to the sender will be echoed back to the screen of the client terminal. In this case, the user terminal usually does not echo characters when they are typed but waits until it receives them from the server.

Suppress go-ahead. This option suppresses the go-ahead (GA) character (see section on Modes of Operation).

Status. This option allows the user or the process running on the client machine to get the status of the options being enabled at the server site.

Timing mark. This option allows one party to issue a timing mark that indicates all previously received data has been processed.

Terminal type. This option allows the client to send its terminal type.

Terminal speed. This option allows the client to send its terminal speed.

Line mode. This option allows the client to switch to the line mode.

Option Negotiation

To use any of the options mentioned in the previous section first requires **option negotiation** between the client and the server. Four control characters are used for this purpose; these are shown in Table –

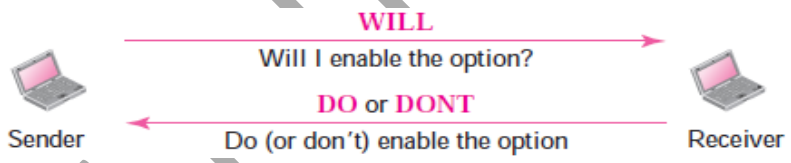
Character	*Code*	*Meaning 1*	*Meaning 2*	*Meaning 3*
WILL	251	Offering to enable	Accepting to enable	
WONT	252	Rejecting to enable	Offering to disable	Accepting to disable
DO	253	Approving to enable	Requesting to enable	
DONT	254	Disapproving to enable	Approving to disable	Requesting to disable

Enabling an Option –

Some options can only be enabled by the server, some only by the client, and some by both. An option is enabled either through an *offer* or a *request*.

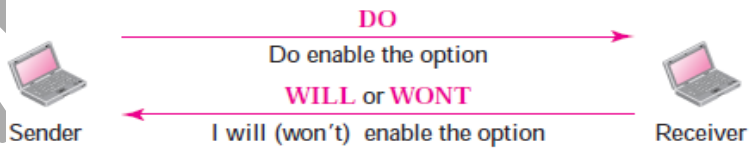
Offer to Enable –

A party can offer to enable an option if it has the right to do so. The offering can be approved or disapproved by the other party. The offering party sends the *WILL* command, which means "Will I enable the option?" The other party sends either the *DO* command, which means "Please do," or the *DONT* command, which means "Please don't." See Figure –



Request to Enable –

A party can request from the other party the enabling of an option. The request can be accepted or refused by the other party. The requesting party sends the *DO* command, which means "Please do enable the option." The other party sends either the *WILL* command, which means "I will," or the *WONT* command, which means "I won't." See Figure –

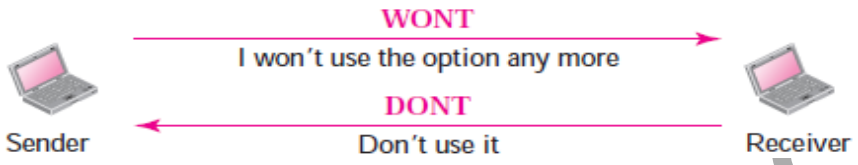


Disabling an Option –

An option that has been enabled can be disabled by one of the parties. An option is disabled either through an *offer* or a *request*.

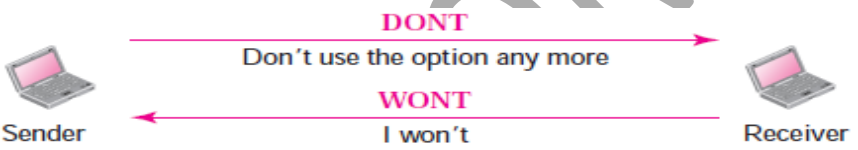
Offer to Disable –

A party can offer to disable an option. The other party must approve the offering; it cannot be disapproved. The offering party sends the *WONT* command, which means "I won't use this option anymore." The answer must be the *DONT* command, which means "Don't use it anymore." Figure – shows an offer to disable an option.

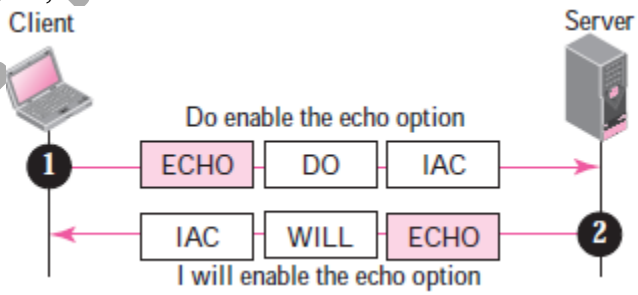


Request to Disable –

A party can request from another party the disabling of an option. The other party must accept the request; it cannot be rejected. The requesting party sends the *DONT* command, which means "Please don't use this option anymore."
The answer must be the *WONT* command, which means "I won't use it anymore." Figure – shows a request to disable an option.



- Figure – shows an example of option negotiation. In this example, the client wants the server to echo each character sent to the server. In other words, when a character is typed at the user keyboard terminal, it goes to the server and is sent back to the screen of the user before being processed.
- The echo option is enabled by the server because it is the server that sends the characters back to the user terminal.
- Therefore, the client should *request* from the server the enabling of the option using DO. The request consists of three characters: IAC, DO, and ECHO. The server accepts the request and enables the option. It informs the client by sending the three character approval: IAC, WILL, and ECHO.



Symmetry –

- One interesting feature of TELNET is its symmetric option negotiation in which the client and server are given equal opportunity. This means that, at the beginning of connection, it is assumed that both sides are using a default TELNET implementation with no options enabled. If one party wants an option enabled, it can offer or request.

- The other party has the right to approve the offer or reject the request if the party is not capable of using the option or does not want to use the option. This allows for the expansion of TELNET.
- A client or server can install a more sophisticated version of TELNET with more options. When it is connected to a party, it can offer or request these new options. If the other party also supports these options, the options can be enabled; otherwise, they are rejected.

Suboption Negotiation –

Some options require additional information. For example, to define the type or speed of a terminal, the negotiation includes a string or a number to define the type or speed. In either case, the two suboption characters indicated in Table – are needed for **suboption negotiation.**

Character	*Decimal*	*Binary*	*Meaning*
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

Controlling the Server –

Some control characters can be used to control the remote server. When an application program is running on the local computer, special characters are used to interrupt (abort) the program (for example, Ctrl+c), or erase the last character typed (for example, delete key or backspace key), and so on. However, when a program is running on a remote computer, these control characters are sent to the remote machine. The user still types the same sequences, but they are changed to special characters and sent to the server.

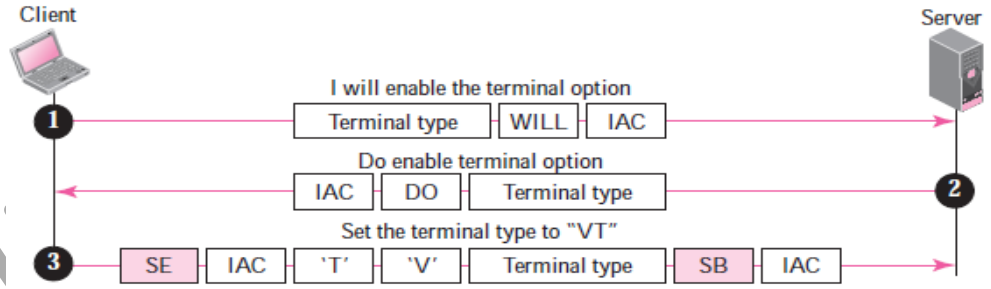


Table – shows some of the characters that can be sent to the server to control the application program that is running there.

Character	*Decimal*	*Binary*	*Meaning*
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase the last character
EL	248	11111000	Erase line

IP (interrupt process) –

- When a program is being run locally, the user can interrupt(abort) the program if, for example, the program has gone into an infinite loop.
- The user can type the Ctrl+c combination, the operating system calls a function, and the function aborts the program. However, if the program is running on a remote machine, the appropriate function should be called by the operating system of the remote machine.
- TELNET defines the IP (interrupt process) control character that is read and interpreted as the appropriate command for invoking the interrupting function in the remote machine.

AO (abort output) –

- This is the same as IP, but it allows the process to continue without creating output. This is useful if the process has another effect in addition to creating output. The user wants this effect but not the output.
- For example, most commands in UNIX generate output and have an exit status. The user may want the exit status for future use but is not interested in the output data.

AYT (are you there?) –

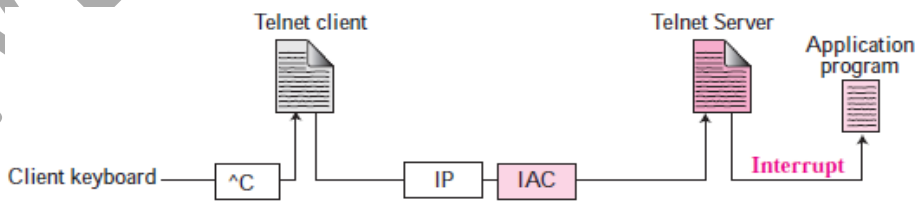
This control character is used to determine if the remote machine is still up and running, especially after a long silence from the server. When this character is received, the server usually sends an audible or visual signal to confirm that it is running.

EC (erase character) –

When a user sends data from the keyboard to the local machine, the delete or backspace character can erase the last character typed. To do the same in a remote machine, TELNET defines the EC control character.

EL (erase line) –

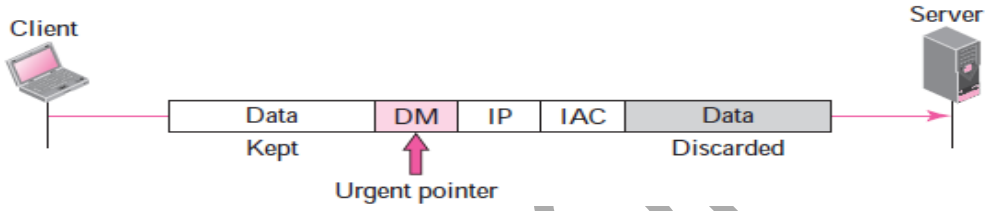
This is used to erase the current line in the remote host.
For example, Figure– shows how to interrupt a runaway application program at the server site. The user types Ctrl+c, but the TELNET client sends the combination of IAC and IP to the server.



Out-of-Band Signaling –

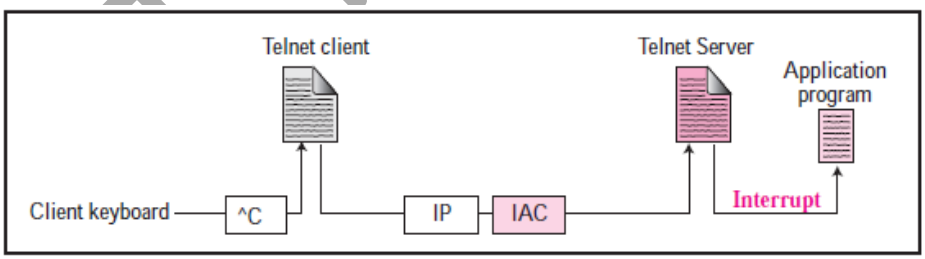
- To make control characters effective in special situations, TELNET uses **out-of-band signaling.** In out-of-band signaling, the control characters are preceded by IAC and are sent to the remote process.

- When a TELNET process (client or server) wants to send an out-of-band sequence of characters to the other process (client or server), it embeds the sequence in the data stream and inserts a special character called a DM (data mark).
- However, to inform the other party, it creates a TCP segment with the urgent bit set and the urgent pointer pointing to the DM character. When the receiving process receives the data, it reads the data and discards any data preceding the control characters (IAC and IP, for example). When it reaches the DM character, the remaining data are handled normally.
- In other words, the DM character is used as a *synchronization* character that switches the receiving process from the urgent mode to the normal mode and *resynchronizes* the two ends (see Figure).
- In this way, the control character (IP) is delivered out of band to the operating system, which uses the appropriate function to interrupt the running application program.

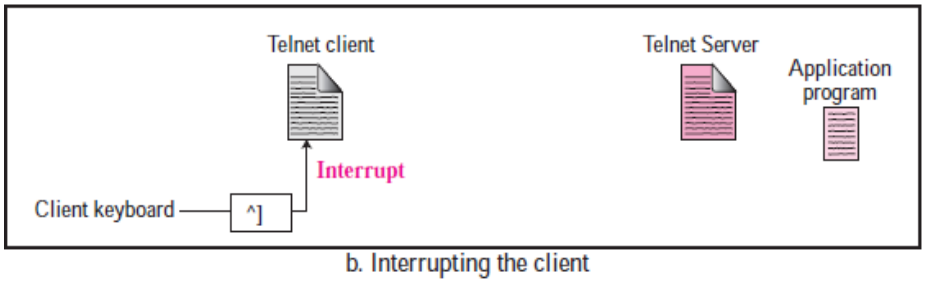


Escape Character –

- A character typed by the user is normally sent to the server. However, sometimes the user wants characters interpreted by the client instead of the server. In this case, the user can use an *escape* character, normally Ctrl+] (shown as ^]).
- Figure –compares the interruption of an application program at the remote site with the interruption of the client process at the local site using the escape character. The TELNET prompt is displayed after this escape character.



a. Interrupting the application program



b. Interrupting the client

Modes of Operation –

Most TELNET implementations operate in one of three modes: default mode, character mode, or line mode.

Default Mode –

- The **default mode** is used if no other modes are invoked through option negotiation. In this mode, the echoing is done by the client. The user types a character and the client echoes the character on the screen (or printer) but does not send it until a whole line is completed.
- After sending the whole line to the server, the client waits for the GA (go ahead) command from the server before accepting a new line from the user.
- The operation is half-duplex. Half-duplex operation is not efficient when the TCP connection itself is full-duplex, and so this mode is becoming obsolete.

Character Mode –

- In the **character mode,** each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed if the transmission time is long (such as in a satellite connection). It also creates overhead (traffic) for the network because three TCP segments must be sent for each character of data:
  1. The user enters a character that is sent to the server.
  2. The server acknowledges the received character and echoes the character back (in one segment).
  3. The client acknowledges the receipt of the echoed character.

Line Mode –

- A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode. In this mode, called the **line mode,** line editing (echoing, character erasing, line erasing, and so on) is done by the client. The client then sends the whole line to the server.
- Although the line mode looks like the default mode, it is not. The default mode operates in the half-duplex mode; the line mode is full-duplex with the client sending one line after another, without the need for an intervening GA(go ahead) character from the server.

User Interface –

- The normal user does not use TELNET commands as defined above. Usually, the operating system (UNIX, for example) defines an interface with user-friendly commands.
- An example of such a set of commands can be found in Table – Note that the interface is responsible for translating the user-friendly commands to the previously defined commands in the protocol.

Security Issue –

- TELNET suffers from security problems. Although TELNET requires a login name and password (when exchanging text), often this is not enough.
- A microcomputer connected to a broadcast LAN can easily eavesdrop using snooper software and capture a login name and the corresponding password (even if it is encrypted).

SECURE SHELL (SSH) –

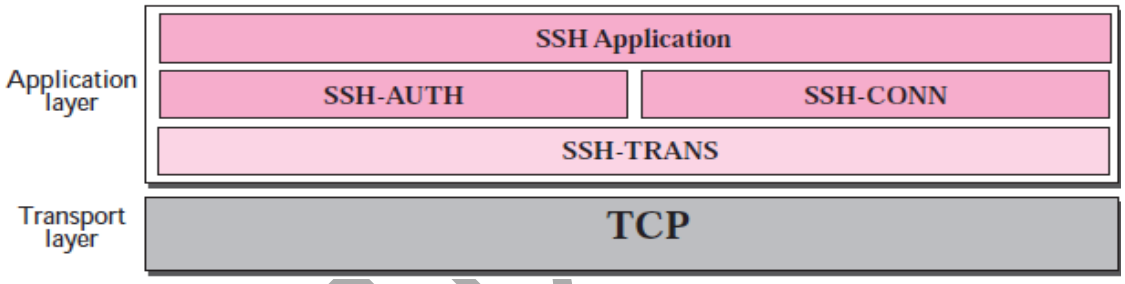
Another popular remote login application program is **Secure Shell (SSH)**. SSH, like TELNET, uses TCP as the underlying transport protocol, but SSH is more secure and provides more services than TELNET.

Versions –

There are two versions of SSH: SSH-1 and SSH-2, which are totally incompatible. The first version, SSH-1 is now deprecated because of security flaws in it.

Components -

SSH is a proposed application-layer protocol with four components, as shown in Figure –



SSH Transport-Layer Protocol (SSH-TRANS)

- Since TCP is not a secured transport layer protocol, SSH first uses a protocol that creates a secured channel on the top of TCP. This new layer is an independent protocol referred to as SSH-TRANS.
- When the software implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure proconnection.
- Then they exchange several security parameters to establish a secure channel on the top of the TCP. we briefly list the services provided by this protocol:

1. Privacy or confidentiality of the message exchanged.

2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder.

3. Server authentication, which means that the client is now sure that the server is the one that it claims to be.

4. Compression of the messages that improve the efficiency of the system and makes attack more difficult.

SSH Authentication Protocol (SSH-AUTH) –

After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another software that can authenticate the client for the server.

SSH Connection Protocol (SSH-CONN) –

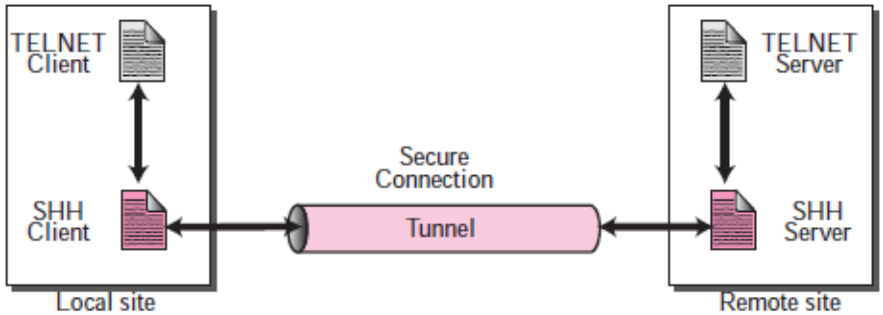
- After the secured channel is established and both server and client are authenticated for each other, SHH can call a piece of software that implements the third protocol, SSHCONN.
- One of the services provided by the SSH-CONN protocol is to do multiplexing. SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it.

SSH Applications –

- After the connection phase is completed, SSH allows several application programs to use the connection. Each application can create a logical channel as described above and then benefit from the secured connection.
- In other words, remote login is one of the services that can use the SSH-CONN protocols; other applications, such as a file transfer application can use one of the logical channels for this purpose.

Port Forwarding –

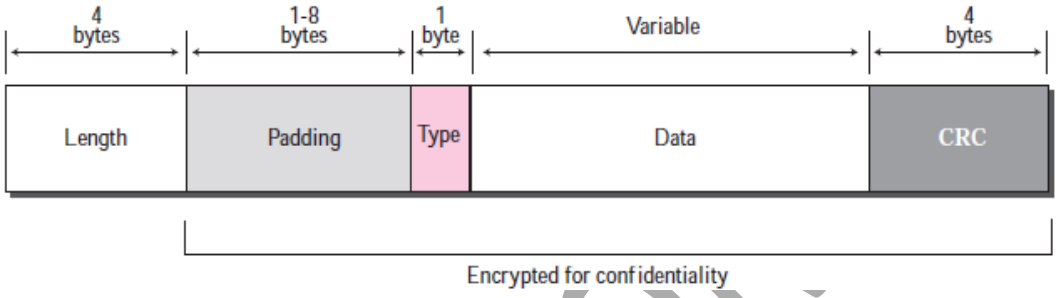
- One of the interesting services provided by the SSH protocol is to provide **port forwarding**. We can use the secured channels available in SSH to access an application program that does not provide security services.
- Application such as TELNET and SMTP can use the services of SSH using port forwarding mechanism.
- SSH port forwarding mechanism creates a tunnel through which the messages belonging to other protocol can travel. For this reason, this mechanism is sometimes referred to as SSH **tunneling**. Figure – shows the concept of port forwarding.



- We can change a direct, but insecure, connection between the TELNET client and the TELNET server by port forwarding. The TELNET client can use the SHH client on the local site to make a secure connection with the SSH server on the remote site.
- Any request from the TELNET client to the TELNET server is carried through the tunnel provided by the SSH client and server. Any response from the TELNET server to the TELNET client is also carried through the tunnel provided by the SSH client and server.

Format of the SSH Packets –

Figure – shows the format of packets used by the SSH protocols.



The following is the brief description of each field:

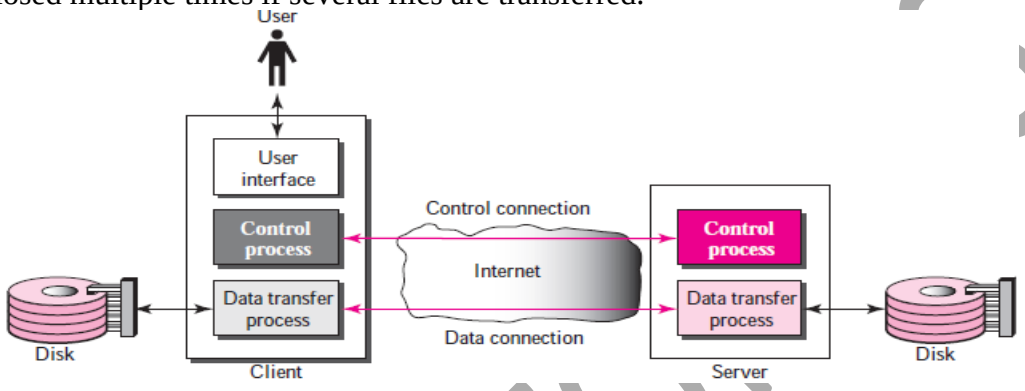
Length. This 4-byte field defines the length of the packet including the type, the data, and the CRC field, but not the padding and the length field.
Padding. One to eight bytes of padding is added to the packet to make the attack on the security provision more difficult.
Type. This one-byte field defines the type of the packet used by SSH protocols.
Data. This field is of variable length. The length of the data can be found by deducting the five bytes from the value of the length field.
CRC. The cyclic redundancy check filed is used for error detection.

FTP –

- **File Transfer Protocol (FTP)** is the standard mechanism provided by TCP/IP for copying a file from one host to another.
- FTP differs from other client-server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses).
- Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication.
- We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.
- FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.
- Figure 21.1 shows the basic model of FTP. The client has three components: user interface, client control process, and the client data transfer process. The server has two

components: the server control process and the server data transfer process. The control connection is made between the control processes.

- The data connection is made between the data transfer processes. The **control connection** remains connected during the entire interactive FTP session.
- The data connection is opened and then closed for each file transferred. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.



Connections

The two FTP connections, control and data, use different strategies and different port numbers.

Control Connection

The control connection is created in the same way as other application programs. There are two steps:

1. The server issues a passive open on the well-known port 21 and waits for a client.
2. The client uses an ephemeral port and issues an active open.

The connection remains open during the entire process. The service type, used by the IP protocol, is *minimize delay* because this is an interactive connection between a user (human) and a server. The user types commands and expects to receive responses without significant delay. Figure – shows the initial connection between the server and the client.

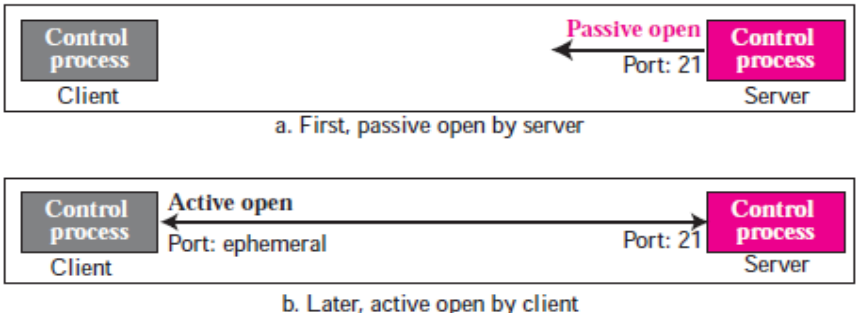


Figure 21.2 *Opening the control connection*

Data Connection –

The **data connection** uses the well-known port 20 at the server site. However, the creation of a data connection is different from what we have seen so far. The following shows how FTP creates a data connection:

1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.
2. The client sends this port number to the server using the PORT command.
3. The server receives the port number and issues an active open using the well known port 20 and the received ephemeral port number. The steps for creating the initial data connection are shown in Figure –

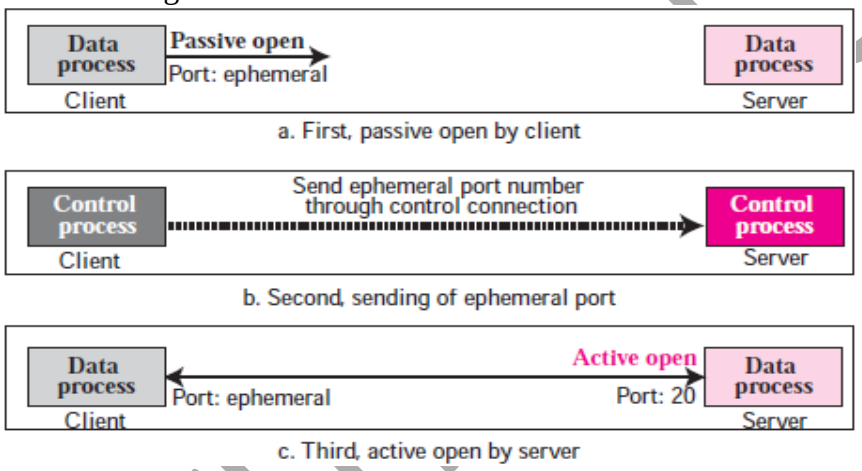


Figure 21.3 *Creating the data connection*

Communication –

The FTP client and server, which run on different computers, must communicate with each other. These two computers may use different operating systems, different character sets, different file structures, and different file formats. FTP must make this heterogeneity compatible. FTP has two different approaches, one for the control connection and one for the data connection.

Communication over Control Connection –

- FTP uses the same approach as TELNET or SMTP to communicate across the control connection. It uses the NVT ASCII character set (see Figure). Communication is achieved through commands and responses.
- This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line so we need not worry about file format or file structure. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.

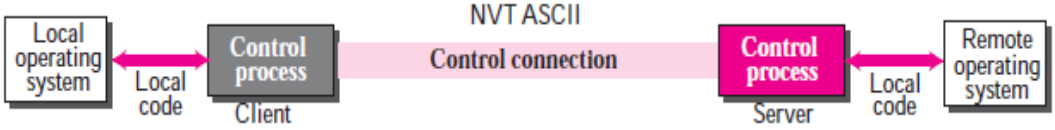


Figure 21.4 *Using the control connection*

Communication over Data Connection –

- The purpose and implementation of the data connection are different from that of the control connection. We want to transfer files through the data connection. The client must define the type of file to be transferred, the structure of the data, and the transmission mode.
- Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode (see Figure).

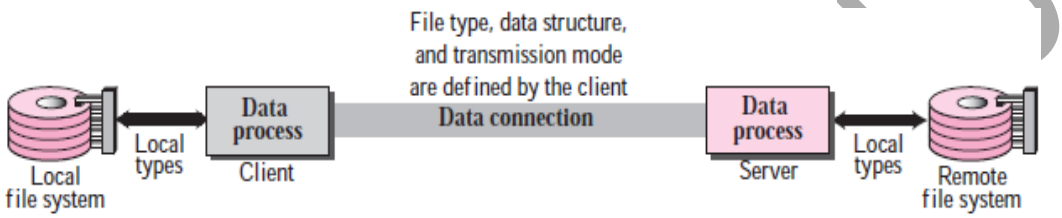


Figure 21.5 *Using the data connection*

File Type

FTP can transfer one of the following file types across the data connection:

- **ASCII file.** This is the default format for transferring text files. Each character is encoded using NVT ASCII. The sender transforms the file from its own representation into NVT ASCII characters and the receiver transforms the NVT ASCII characters to its own representation.
- **EBCDIC file.** If one or both ends of the connection use EBCDIC encoding, the file can be transferred using EBCDIC encoding.
- **Image file.** This is the default format for transferring binary files. The file is sent as continuous streams of bits without any interpretation or encoding. This is mostly used to transfer binary files such as compiled programs.
  If the file is encoded in ASCII or EBCDIC, another attribute must be added to define the printability of the file.

a. **Nonprint.** This is the default format for transferring a text file. The file contains no vertical specifications for printing. This means that the file cannot be printed without further processing because there are no characters to be interpreted for vertical movement of the print head. This format is used for files that will be stored and processed later.
b. **TELNET.** In this format the file contains NVT ASCII vertical characters such as CR (carriage return), LF (line feed), NL (new line), and VT (vertical tab). The file is printable after transfer.

Transmission Mode –

FTP can transfer a file across the data connection using one of the following three transmission modes:

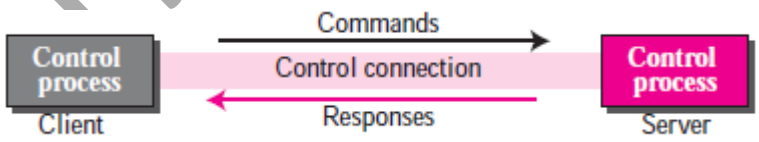
1. **Stream mode.** This is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data is simply a stream of bytes (file structure), no end-offile is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a 1-byte end-of-record (EOR) character and the end of the file will have a 1-byte end-of-file (EOF) character.
2. **Block mode.** Data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the *block descriptor;* the next two bytes define the size of the block in bytes.
3. **Compressed mode.** If the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the number of repetitions. In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed.

Data Structure –

FTP can transfer a file across the data connection using one of the following interpretations about the structure of the data:
File structure (default) – The file has no structure. It is a continuous stream of bytes.
Record structure – The file is divided into records. This can be used only with text files.
Page structure – The file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Command Processing –

FTP uses the control connection to establish a communication between the client control process and the server control process. During this communication, the commands are sent from the client to the server and the responses are sent from the server to the client in the Figure –



Commands –

Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. We can roughly divide the commands into six groups: access commands, file management commands, data formatting commands, port defining commands, file transferring commands, and miscellaneous commands.

Access commands. These commands let the user access the remote system. Table lists common commands in this group.

Command	*Argument(s)*	*Description*
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command

File management commands. These commands let the user access the file system on the remote computer. They allow the user to navigate through the directory structure, create new directories, delete files, and so on. Table gives common commands in this group.

Command	*Argument(s)*	*Description*
CWD	Directory name	Change to another directory
CDUP		Change to parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List subdirectories or files without attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed
RNTO	File name (new)	Rename the file
SMNT	File system name	Mount a file system

Data formatting commands. These commands let the user define the data structure, file type, and transmission mode. The defined format is then used by the file transfer commands. Table – shows common commands in this group.

Command	*Argument(s)*	*Description*
TYPE	A (ASCII), E (EBCDIC), I (Image), N (Nonprint), or T (TELNET)	Define file type
STRU	F (File), R (Record), or P (Page)	Define organization of data
MODE	S (Stream), B (Block), or C (Compressed)	Define transmission mode

Port defining commands.

These commands define the port number for the data connection on the client site. There are two methods to do this. In the first method, using the PORT command, the client can choose an ephemeral port number and send it to the server using a passive open. The server uses that port number and creates an active open. In the second method, using the PASV command, the client just asks the server to first choose a port number. The server does a passive open on that port and sends the port number in the response (see response numbered 227 in Table). The client issues an active open using that port number. Table shows the port defining commands.

Command	*Argument(s)*	*Description*
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

File transfer commands. These commands actually let the user transfer files. Table 21.5 lists common commands in this group.

Command	*Argument(s)*	*Description*
RETR	File name(s)	Retrieve files; file(s) are transferred from server to client
STOR	File name(s)	Store files; file(s) are transferred from client to server
APPE	File name(s)	Similar to STOR, but if file exists, data must be appended to it
STOU	File name(s)	Same as STOR, but file name will be unique in the directory
ALLO	File name(s)	Allocate storage space for files at the server
REST	File name(s)	Position file marker at a specified data point
STAT	File name(s)	Return status of files

Miscellaneous commands. These commands deliver information to the FTP user at the client site. Table shows common commands in this group.

Command	*Argument(s)*	*Description*
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server

Responses –

Every FTP command generates at least one response. A response has two parts: a three digit number followed by text. The numeric part defines the code; the text part defines needed parameters or extra explanations. We represent the three digits as *xyz*. The meaning of each digit is described below.

First Digit – The first digit defines the status of the command. One of five digits can be used in this position:

1yz (positive preliminary reply). The action has started. The server will send another reply before accepting another command.

2yz (positive completion reply). The action has been completed. The server will accept another command.

3yz (positive intermediate reply). The command has been accepted, but further information is needed.

4yz (transient negative completion reply). The action did not take place, but the error is temporary. The same command can be sent later.

5yz (permanent negative completion reply). The command was not accepted and should not be retried again.

Second Digit – The second digit also defines the status of the command. One of six digits can be used in this position:

x0z (syntax).

x1z (information).

x2z (connections).

x3z (authentication and accounting).

x4z (unspecified).
x5z (file system).
Third Digit – The third digit provides additional information. Table – shows a brief list of possible responses (using all three digits).

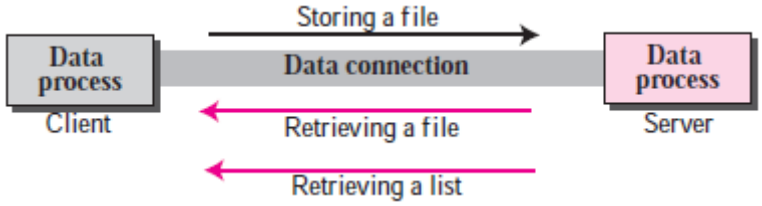
Code	*Description*
Positive Preliminary Reply	
120	Service will be ready shortly
125	Data connection open; data transfer will start shortly
150	File status is OK; data connection will be open shortly
Positive Completion Reply	
200	Command OK
211	System status or help reply
212	Directory status
213	File status
214	Help message
215	Naming the system type (operating system)
220	Service ready
221	Service closing
225	Data connection open
226	Closing data connection
227	Entering passive mode; server sends its IP address and port number
230	User login OK
250	Request file action OK
Positive Intermediate Reply	
331	User name OK; password is needed
332	Need account for logging
350	The file action is pending; more information needed

Code	*Description*
Transient Negative Completion Reply	
425	Cannot open data connection
426	Connection closed; transfer aborted
450	File action not taken; file not available
451	Action aborted; local error
452	Action aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
530	User not logged in
532	Need account for storing file
550	Action is not done; file unavailable
552	Requested action aborted; exceeded storage allocation
553	Requested action not taken; file name not allowed

File Transfer

File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things (see Figure).

- A file is to be copied from the server to the client (download). This is called *retrieving a file*. It is done under the supervision of the RETR command.
- A file is to be copied from the client to the server (upload). This is called *storing a file*. It is done under the supervision of the STOR command.
- A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command. Note that FTP treats a list of directory or file names as a file. It is sent over the data connection.



TFTP

There are occasions when we need to simply copy a file without the need for all of the features of the FTP protocol. For example, when a diskless workstation or a router is booted, we need to download the bootstrap and configuration files. Here we do not need all of the sophistication provided in FTP. We just need a protocol that quickly copies the files.

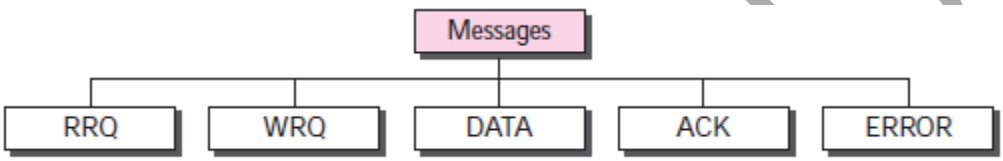
Trivial File Transfer Protocol (TFTP) is designed for these types of file transfer. It is so simple that the software package can fit into the read-only memory of a diskless workstation. It

can be used at bootstrap time. The reason that it fits on ROM is that it requires only basic IP and UDP. However, there is no security for TFTP. TFTP can read or write a file for the client. *Reading* means copying a file from the server site to the client site. *Writing* means copying a file from the client site to the server site.

TFTP uses the services of UDP on the well-known port 69.

Messages –

There are five types of TFTP messages, RRQ, WRQ, DATA, ACK, and ERROR, as shown in Figure –



RRQ –

The read request (RRQ) message is used by the client to establish a connection for reading data from the server. Its format is shown in Figure –

The RRQ message fields are as follows:

OpCode – The first field is a 2-byte operation code. The value is 1 for the RRQ message.

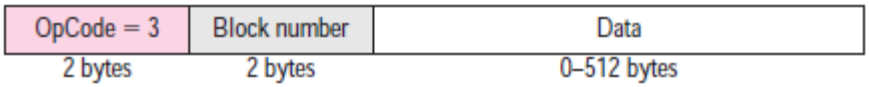
File name – The next field is a variable-size string (encoded in ASCII) that defines the name of the file. Since the file name varies in length, termination is signaled by a 1-byte field of 0s.

Mode – The next field is another variable-size string defining the transfer mode. The mode field is terminated by another 1-byte field of 0s. The mode can be one of two strings: "netascii" (for an ASCII file) or "octet" (for a binary file). The file name and mode fields can be in upper- or lowercase, or a combination of both.

WRQ –
The write request (WRQ) message is used by the client to establish a connection for writing data to the server. The format is the same as RRQ except that the OpCode is 2 in the Figure –

DATA
The data (DATA) message is used by the client or the server to send blocks of data. Its format is shown in Figure –. The DATA message fields are as follows:

OpCode – The first field is a 2-byte operation code. The value is 3 for the DATA message.



Block number. This is a 2-byte field containing the block number. The sender of the data (client or server) uses this field for sequencing. All blocks are numbered sequentially starting with 1. The block number is necessary for acknowledgment.

Data. This block must be exactly 512 bytes in all DATA messages except the last block, which must be between 0 and 511 bytes. A non-512 byte block is used as a signal that the sender has sent all the data. In other words, it is used as an end-of-file indicator. If the data in the file happens to be an exact multiple of 512 bytes, the sender must send one extra block of zero bytes to show the end of transmission. Data can be transferred in either NVT ASCII (netascii) or binary octet (octet).

ACK –

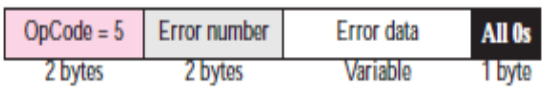
The acknowledge (ACK) message is used by the client or server to acknowledge the receipt of a data block. The message is only 4 bytes long. Its format is shown in Figure –

The ACK message fields are as follows:

OpCode. The first field is a 2-byte operation code. The value is 4 for the ACK message.
Block number. The next field is a 2-byte field containing the number of the block received.
The ACK message can also be a response to a WRQ. It is sent by the server to indicate that it is ready to receive data from the client. In this case the value of the block number field is 0.

ERROR –

The ERROR message is used by the client or the server when a connection cannot be established or when there is a problem during data transmission. It can be sent as a negative response to RRQ or WRQ. It can also be used if the next block cannot be transferred during the actual data transfer phase. The error message is not used to declare a damaged or duplicated message. These problems are resolved by error-control mechanisms. The format of the ERROR message is shown in Figure –



The ERROR message fields are as follows:

OpCode – The first field is a 2-byte operation code. The value is 5 for the ERROR message.

Error number – This 2-byte field defines the type of error. Table – shows the error numbers and their corresponding meanings.

Number	*Meaning*	*Number*	*Meaning*
0	Not defined	5	Unknown port number
1	File not found	6	File already exists
2	Access violation	7	No such user
3	Disk full or quota exceeded		
4	Illegal operation		

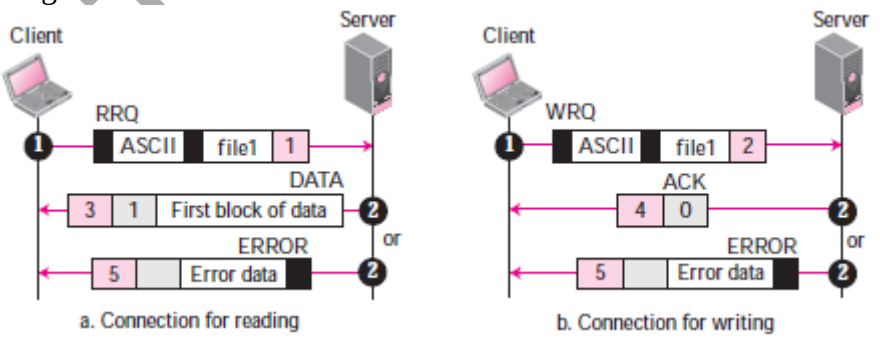
Error data – This variable-byte field contains the textual error data and is terminated by a 1-byte field of 0s.

Connection –

- TFTP uses UDP services. Because there is no provision for connection establishment and termination in UDP, UDP transfers each block of data encapsulated in an independent user datagram.
- In TFTP, however, we do not want to transfer only one block of data; we do not want to transfer the file as independent blocks either. We need connections for the blocks of data being transferred if they all belong to the same file.
- TFTP uses RRQ, WRQ, ACK, and ERROR messages to establish connection. It uses the DATA message with a block of data of fewer than 512 bytes (0–511) to terminate connection.

***Connection Establishment* –**

Connection establishment for reading files is different from connection establishment for writing files in the Figure –



Reading –

To establish a connection for **reading,** the TFTP client sends the RRQ message. The name of the file and the transmission mode is defined in this message.
If the server can transfer the file, it responds positively with a DATA message containing the first block of data. If there is a problem, such as difficulty in opening the file or permission restriction, the server responds negatively by sending an ERROR message.

Writing –

To establish a connection for **writing,** the TFTP client uses the WRQ message. The name of the file and the transmission mode is defined in this message.
If the server can accept a copy of the file, it responds positively with an ACK message using a value of 0 for the block number. If there is any problem, the server responds negatively by sending an ERROR message.

Connection Termination –

After the entire file is transferred, the connection must be terminated. TFTP does not have a special message for termination. Termination is accomplished by sending the last block of data, which is less than 512 bytes.

Data Transfer –

The data transfer phase occurs between connection establishment and termination. TFTP uses the services of UDP, which is unreliable.
The file is divided into blocks of data, in which each block except the last one is exactly 512 bytes. The last block must be between 0 and 511 bytes. TFTP can transfer data in ASCII or binary format.
UDP does not have any mechanism for flow and error control. TFTP has to create a flow- and error-control mechanism to transfer a file made of continuous blocks of data.

Flow Control –

TFTP sends a block of data using the DATA message and waits for an ACK message. If the sender receives an acknowledgment before the time-out, it sends the next block.
Thus, **flow control** is achieved by numbering the data blocks and waiting for an ACK before the next data block is sent.

Retrieve a File –

When the client wants to retrieve (read) a file, it sends the RRQ message. The server responds with a DATA message sending the first block of data (if there is no problem) with a block number of 1.

Store a File –

When the client wants to store (write) a file, it sends the WRQ message. The server responds with an ACK message (if there is no problem) using 0 for the block number. After receiving this acknowledgment, the client sends the first data block with a block number of 1.

Error Control –

The TFTP error-control mechanism is different from those of other protocols. It is *symmetric,* which means that the sender and the receiver both use time-outs. The sender uses a time-out for data messages; the receiver uses a time-out for acknowledgment
messages. If a data message is lost, the sender retransmits it after time-out expiration. If an acknowledgment is lost, the receiver retransmits it after time-out expiration. This guarantees a smooth operation.

Error control is needed in four situations: a damaged message, a lost message, a lost acknowledgment, or a duplicated message.

Damaged Message There is no negative acknowledgment. If a block of data is damaged, it is detected by the receiver and the block is discarded. The sender waits for the acknowledgment and does not receive it within the time-out period. The block is then sent again. Note that there is no checksum field in the DATA message of TFTP. The only way the receiver can detect data corruption is through the checksum field of the UDP user datagram.

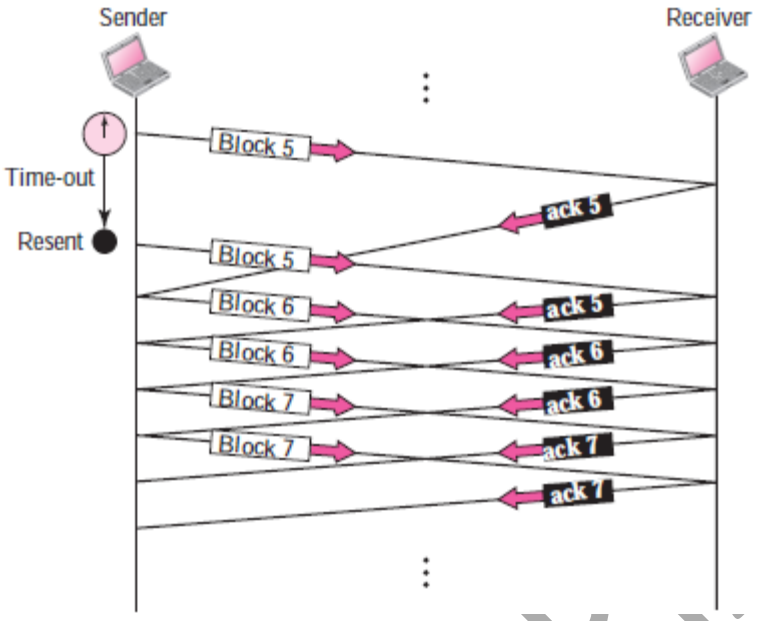
Lost Message If a block is lost, it never reaches the receiver and no acknowledgment is sent. The sender resends the block after the time-out.

Lost Acknowledgment If an acknowledgment is lost, we can have two situations. If the timer of the receiver matures before the timer of the sender, the receiver retransmits the acknowledgment; otherwise, the sender retransmits the data.

Duplicate Message Duplication of blocks can be detected by the receiver through block number. If a block is duplicated, it is simply discarded by the receiver.

Sorcerer's Apprentice Bug –
Although the flow- and error-control mechanism is symmetric in TFTP, it can lead to a problem known as the **sorcerer's apprentice bug,** named for the cartoon character who inadvertently conjures up a mop that continuously replicates itself. This will happen if the ACK message for a packet is not lost but delayed. In this situation, every succeeding block is sent twice and every succeeding acknowledgment is received twice.

Figure – shows this situation. The acknowledgment for the fifth block is delayed. After the time-out expiration, the sender retransmits the fifth block, which will be acknowledged by the receiver again. The sender receives two acknowledgments for the fifth block, which triggers it to send the sixth block twice. The receiver receives the sixth block twice and again sends two acknowledgments, which results in sending the seventh block twice. And so on.



UDP Ports –

When a process uses the services of UDP, the server process issues a passive open on the well-known port and waits for the client process to issue an active open on an ephemeral port. After the connection is established, the client and server communicate using these two ports.

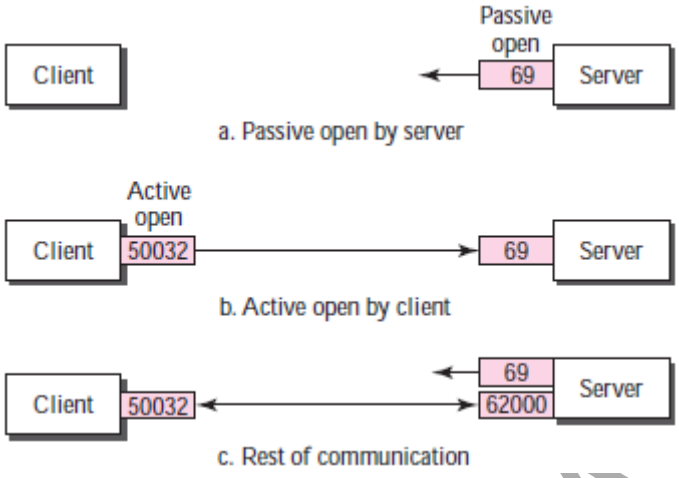
TFTP follows a different set of steps because the communication between a client TFTP and a server TFTP can be quite lengthy (seconds or even minutes). If a TFTP server uses the well-known port 69 to communicate with a single client, no other clients

can use these services during that time. The solution to this problem, as shown in Figure – is to use the well-known port for the initial connection and an ephemeral port for the remaining communication.

The steps are as follows:

1. The server passively opens the connection using the well-known port 69.

2. A client actively opens a connection using an ephemeral port for the source port and the well-known port 69 for the destination port. This is done through the RRQ message or the WRQ message.

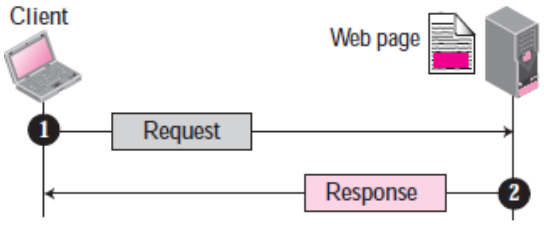


3. The server actively opens a connection using a new ephemeral port for the source port and uses the ephemeral port received from the client as the destination port. It sends the DATA or ACK or ERROR message using these ports. This frees the wellknown port (69) for use by other clients. When the client receives the first message from the server, it uses its own ephemeral port and the ephemeral port sent by the server for future communication.

World Wide Web and HTTP –
ARCHITECTURE –

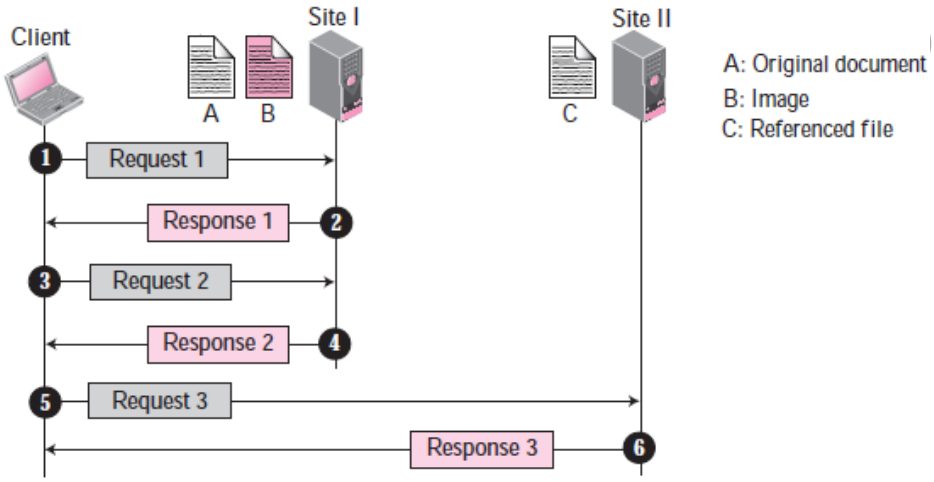
- The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites.*
- Each site holds one or more documents, referred to as Web pages. Each **Web page**, however, can contain some links to other Web pages in the same or other sites. In other words, a Web page can be simple or composite.
- A simple Web page has no link to other Web pages; a composite Web page has one or more links to other Web pages. Each Web page is a file with a name and address.

- Assume we need to retrieve a Web page that contains the biography of a famous character with some pictures, which are embedded in the page itself. Since the pictures are not stored as separate files, the whole document is a simple Web page. It can be retrieved using one single request/ response transaction, as shown in Figure –



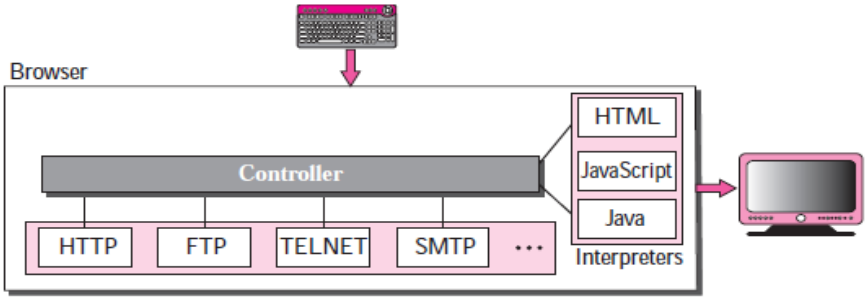
- Now assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image. Figure – shows the situation. The main document and the image are stored in two separate files in the same site (file A and file B); the referenced text file is stored in another site (file C).

- Since we are dealing with three different files, we need three transactions if we want to see the whole document. The first transaction (request/response) retrieves a copy of the main document (file A), which has a reference (pointer) to the second and the third files.
- When a copy of the main document is retrieved and browsed, the user can click on the reference to the image to invoke the second transaction and retrieve a copy of the image (file B). If the user further needs to see the contents of the referenced text file, she can click on its reference (pointer) invoking the third transaction and retrieving a copy of the file C.
- Note that although files A and B both are stored in site I, they are independent files with different names and addresses. Two transactions are needed to retrieve them.



Web Client (Browser)

A variety of vendors offer commercial **browsers** that interpret and display a Web document, and all of them use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters as shown figure –



- The controller receives input from the keyboard or the mouse and uses the client programs to access the document.
- After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as FTP, or TELNET, or HTTP.
- The interpreter can be HTML, Java, or JavaScript, depending on the type of document. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

Web Server –

- The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk.
- A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time. Some popular Web servers include Apache and Microsoft Internet Information Server.

Uniform Resource Locator (URL) –

A client that wants to access a Web page needs the file name and the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The **uniform resource locator (URL)** is a standard locator for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path as shown in the figure –

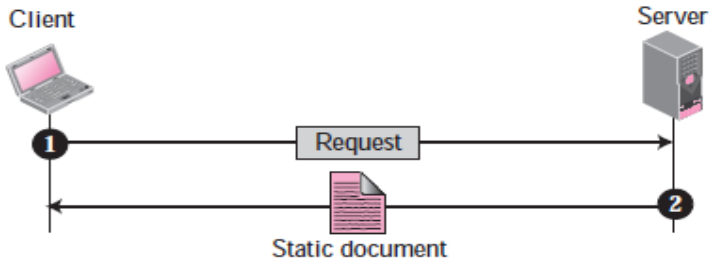
- The *protocol* is the client-server application program used to retrieve the document. Many different protocols can retrieve a document; among them are Gopher, FTP, HTTP, News, and TELNET. The most common today is HTTP.
- The **host** is the domain name of the computer on which the information is located. Web pages are usually stored in computers, and computers are given domain name aliases that usually begin with the characters "www". This is not mandatory, however, as the host can have any domain name.
- The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.
- **Path** is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files. In other words, the path defines the complete file name where the document is stored in the directory system.

WEB DOCUMENTS –

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static Documents –

- **Static documents** are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. In other words, the contents of the file are determined when the file is created, not when it is used.
- Of course, the contents in the server can be changed, but the user cannot change them. When a client accesses the document, a copy of the document is sent. The user can then use a browsing program to display the document (see Figure).



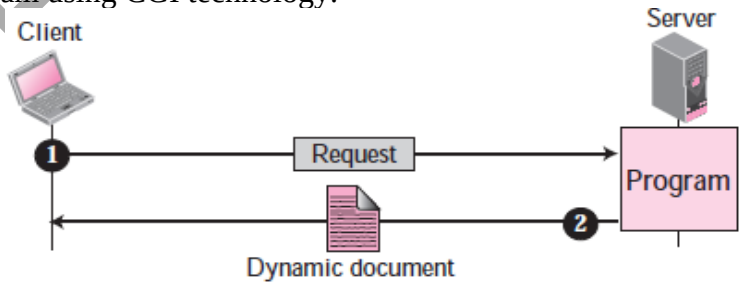
Static documents are prepared using one of the several languages: **Hypertext Markup Language (HTML)**, **Extensible Markup Language (XML)**, **Extensible Style Language (XSL)**, and **Extended Hypertext Markup Language (XHTML)**.

Dynamic Documents –

- A **dynamic document** is created by a Web server whenever a browser requests the document. When a request arrives, the Web server runs an application program or a script that creates the dynamic document.
- The server returns the output of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server.
- Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the *date* program in UNIX and send the result of the program to the client.

Common Gateway Interface (CGI) –

- The **Common Gateway Interface (CGI)** is a technology that creates and handles dynamic documents. CGI is a set of standards that defines how a dynamic document is written, how data are input to the program, and how the output result is used.
- The term *common* in CGI indicates that the standard defines a set of rules that is common to any language or platform.
- The term *gateway* here means that a CGI program can be used to access other resources such as databases, graphic packages, and so on.
- The term *interface* here means that there is a set of predefined terms, variables, calls, and so on that can be used in any CGI program. Figure – illustrates the steps in creating a dynamic program using CGI technology.



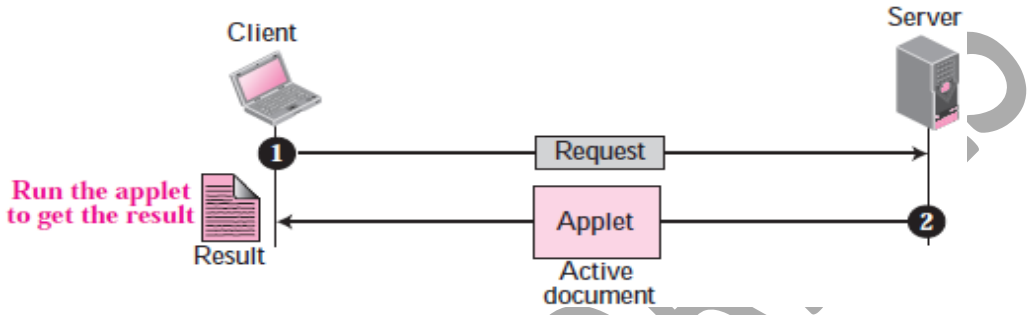
Active Documents –

For many applications, we need a program or a script to be run at the client site. These are called **active documents.** For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user.

The program definitely needs to be run at the client site where the animation or interaction takes place. When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client (browser) site.

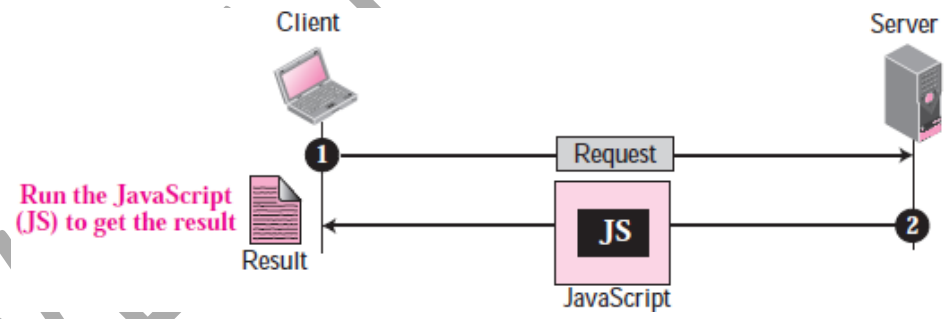
Java Applets –

One way to create an active document is to use **Java applets. Java** is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document (an applet) and a browser to run it. It can also be a stand-alone program that doesn't use a browser. Figure – shows how Java applets.



JavaScript –

- The idea of scripts in dynamic documents can also be used for active documents. If the active part of the document is small, it can be written in a scripting language; then it can be interpreted and run by the client at the same time. The script is in source code (text) and not in binary form.
- The scripting technology used in this case is usually JavaScript. **JavaScript,** which bears a small resemblance to Java, is a very high level scripting language developed for this purpose. Figure – shows how JavaScript is used to create an active document.



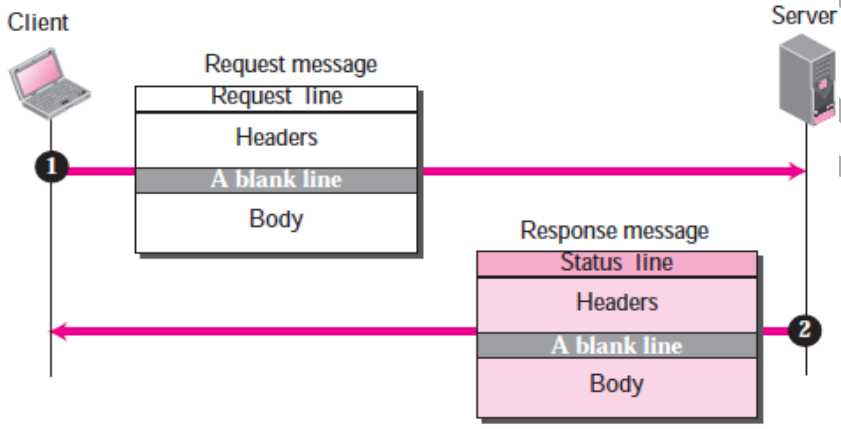
HTTP

- The **Hypertext Transfer Protocol (HTTP)** is a protocol used mainly to access data on the World Wide Web. HTTP functions like a combination of FTP and SMTP. It is similar to FTP because it transfers files and uses the services of TCP.
- However, it is much simpler than FTP because it uses only one TCP connection. There is no separate control connection; only data are transferred between the client and the server.
- HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser).

- SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP Transaction –

Figure – illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol, which means that the server does not keep information about the client. The client initializes the transaction by sending a request. The server replies by sending a response.

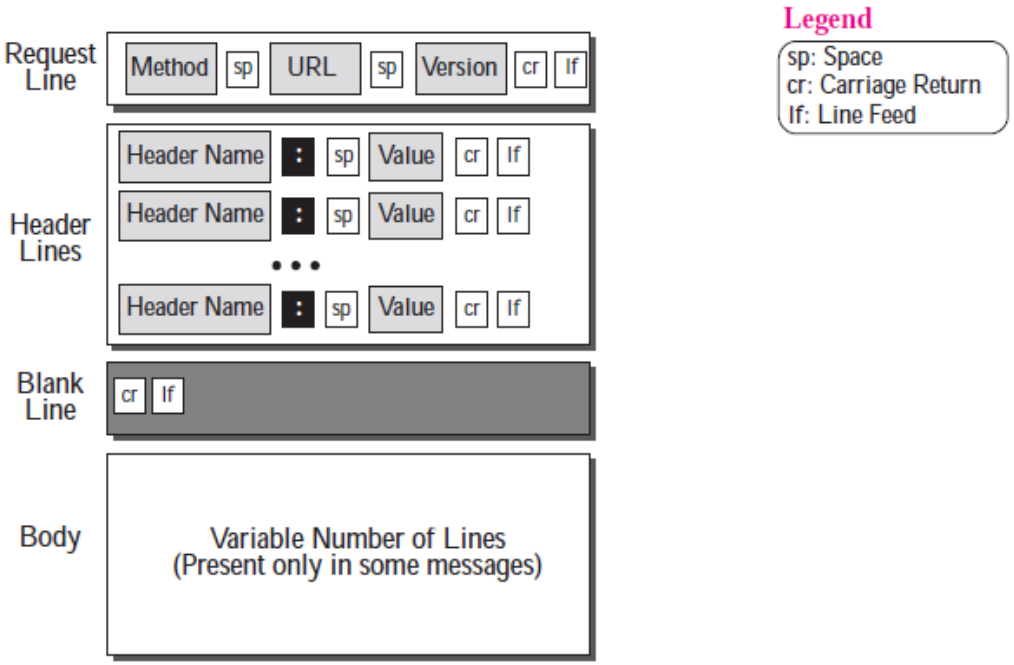


Request Message –

The format of the request is shown in Figure. A request message consists of a request line, a header, and sometimes a body.

Request Line –

The first line in a request message is called a **request line.** There are three fields in this line separated by some character delimiter as shown in Figure. The fields are called methods, URL, and Version. These three should be separated by a space character. At the end two characters, a carriage return followed by a line feed, terminate the line.



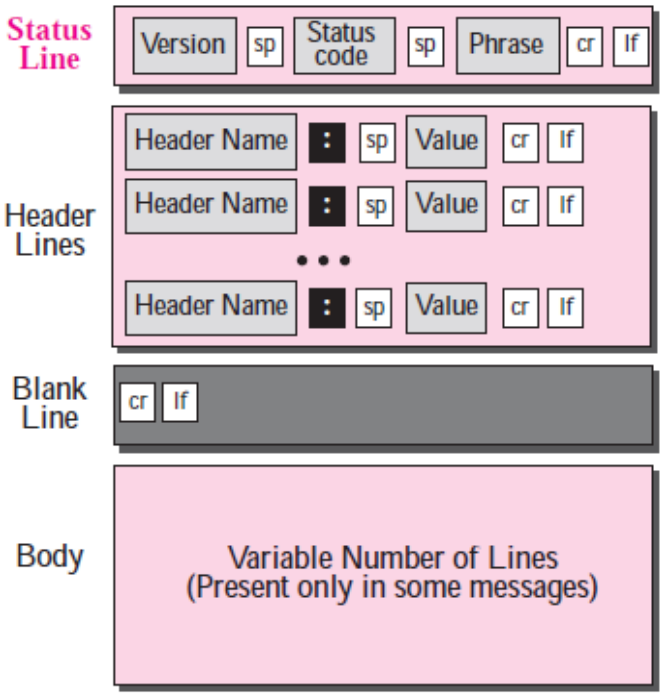
Header Lines In Request Message –
After the request line, we can have zero or more **request header** lines. Each header line sends additional information from the client to the server. For example, the client can request that the document be sent in a special format. Each header line has a header name, a colon, a space, and a header value.

Body In Request Message –
The body can be present in a request message. Usually, it contains the comment to be sent.

Response Message –
The format of the response message is shown in Figure – A response message consists of a status line, header lines, a blank line and sometimes a body.

Status Line –
The first line in a response message is called the **status line**. There are three fields in this line separated by spaces and terminated by a carriage return and line feed. The first field defines the version of HTTP protocol, currently 1.1. The status code field defines the status of the request. The status phrase explains the status code in text form.

Header Lines In Response Message –
After the status line, we can have zero or more **response header** lines. Each header line sends additional information from the server to the client. For example, the sender can send extra information about the document.



Body –

The body contains the document to be sent from the server to the client. The body is present unless the response is an error message.

UNIT VI

Electronic Mail: SMTP, POP, IMAP, and MIME

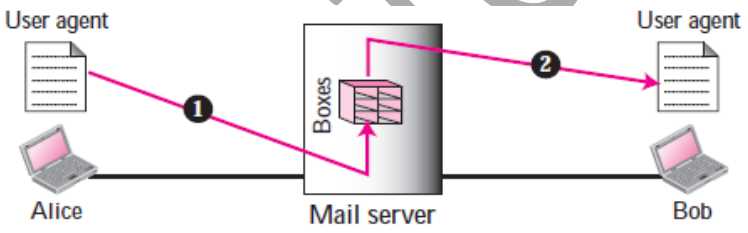
One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program.

ARCHITECTURE –

To explain the architecture of e-mail, we give four scenarios.The fourth scenario is the most common in the exchange of e-mail.

First Scenario –

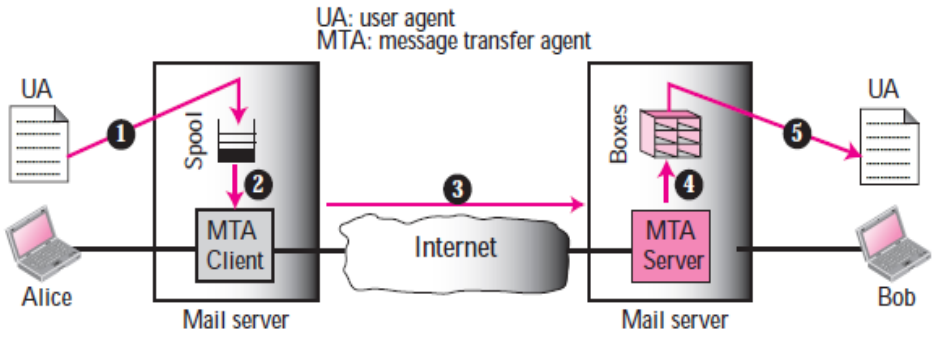
- In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same mail server; they are directly connected to a shared mail server. The administrator has created one mailbox for each user where the received messages are stored.
- A *mailbox* is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice needs to send a message to Bob, she runs a *user agent* (*UA*) program to prepare the message and store it in Bob's mailbox.
- The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience using a user agent. Figure – shows the concept.



- This is similar to the traditional memo exchange between employees in an office. There is a mail room where each employee has a mailbox with his or her name on it.
- When Alice needs to send a memo to Bob, she writes the memo and inserts it into Bob's mailbox. When Bob checks his mailbox, he finds Alice's memo and reads it.
- **When the sender and the receiver of an e-mail are on the same mail server, we need only two user agents.**

Second Scenario –

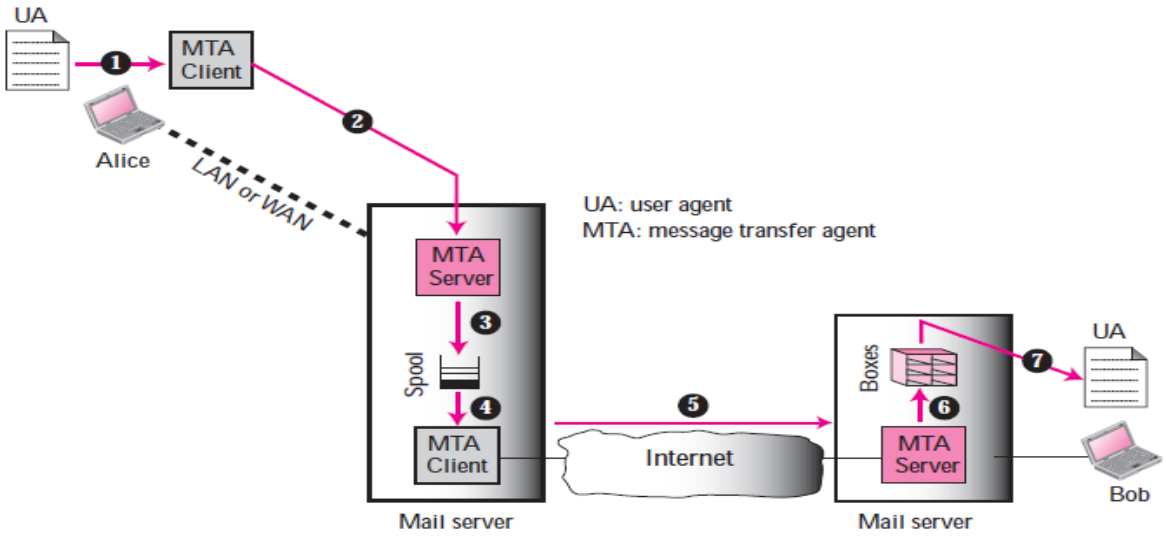
- In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different mail servers. The message needs to be sent over the Internet. Here we need **user agents (UAs)** and **message transfer agents (MTAs)** as shown in Figure –



- Alice needs to use a user agent program to send her message to the mail server at her own site. The mail server at her site uses a queue (spool) to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site.
- The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one client and one server. Like most client-server programs on the Internet, the server needs to run all of the time because it does not know when a client will ask for a connection.
- The client, on the other hand, can be triggered by the system when there is a message in the queue to be sent.
- **When the sender and the receiver of an e-mail are on different mail servers, we need two UAs and a pair of MTAs (client and server).**

Third Scenario

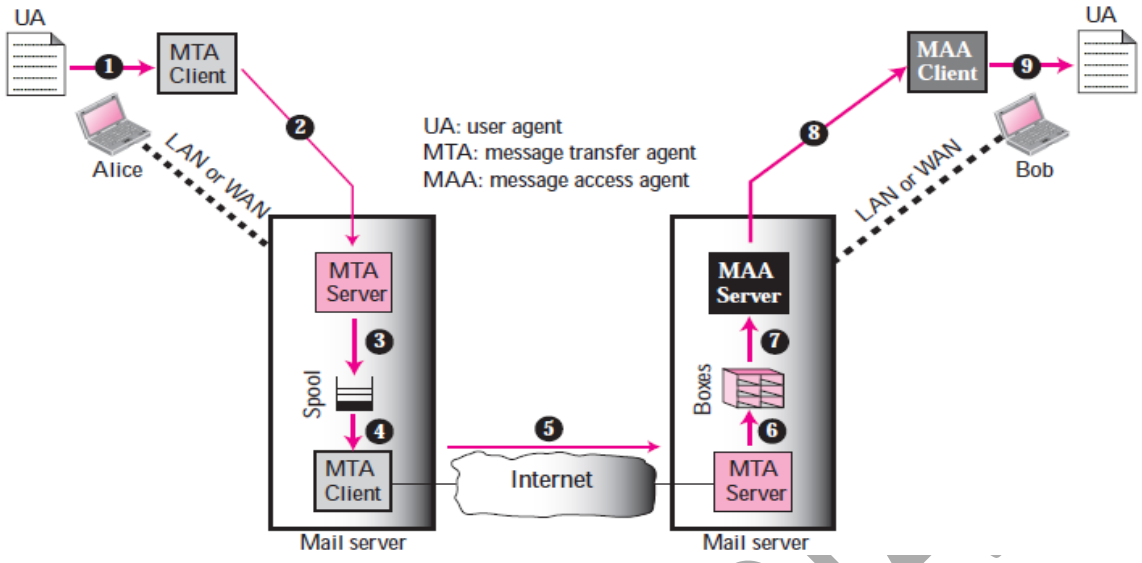
- Figure – shows the third scenario. Bob, as in the second scenario, is directly connected to his mail server. Alice, however, is separated from her mail server. Alice is either connected to the mail server via a point-to-point WAN—such as a dial-up modem, a DSL, or a cable modem—or she is connected 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.
- Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in turn, calls the MTA client.



- The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox.
- At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client-server programs.
- **When the sender is connected to the mail server via a LAN or a WAN, we need two 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. The situation is shown in Figure –



- **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). This is the most common situation today.**

USER AGENT –

The first component of an electronic mail system is the **user agent (UA).** It provides service to the user to make the process of sending and receiving a message easier.

Services Provided by a User Agent

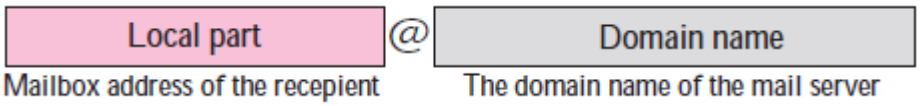
A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers.

User Agent Types –

- There are two types of user agents: command-driven and GUI-based. Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one character command from the keyboard to perform its task.
- For example, a user can type the character *r*, at the command prompt, to reply to the sender of the message, or type the character *R* to reply to the sender and all recipients.
- **Some examples of command-driven user agents are *mail, pine,* and *elm*.**
- 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, Outlook,* and *Netscape*.**

Addresses –

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a **local part** and a **domain name,** separated by an @ sign (see Figure).



Local Part –

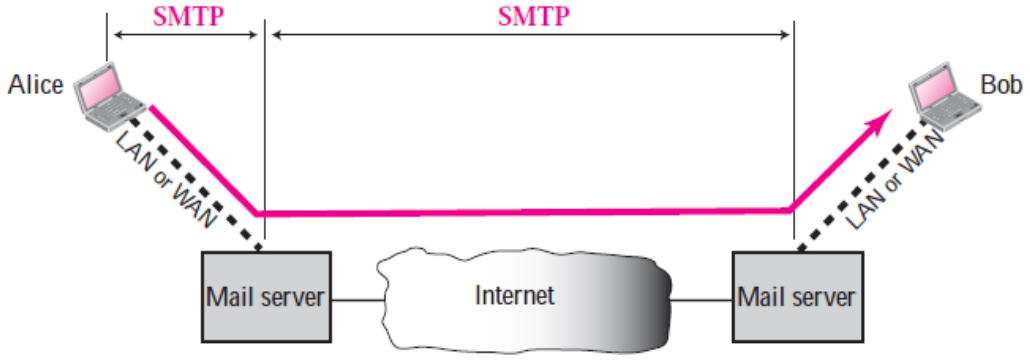
The local part defines the name of a special file, called the user mailbox, where all of the mail received for a user is stored for retrieval by the message access agent.

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; they are sometimes called *mail servers* or *exchangers*. The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name.

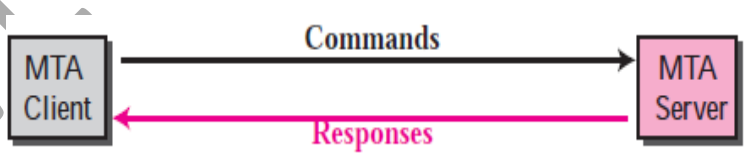
MESSAGE TRANSFER AGENT: SMTP

- The actual mail transfer is done through message transfer agents (MTAs). 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 **Simple Mail Transfer Protocol (SMTP). T**wo pairs of MTA client/server programs are used in the most common situation (fourth scenario). Figure – shows the range of the SMTP protocol in this scenario.



Commands and Responses –

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server (see Figure).



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

Commands –

Commands are sent from the client to the server. The format of a command is shown below:

Keyword: argument(s)

It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands listed in Table

Keyword	*Argument(s)*	*Keyword*	*Argument(s)*
HELO	Sender's host name	NOOP	
MAIL FROM	Sender of the message	TURN	
RCPT TO	Intended recipient	EXPN	Mailing list
DATA	Body of the mail	HELP	Command name
QUIT		SEND FROM	Intended recipient
RSET		SMOL FROM	Intended recipient
VRFY	Name of recipient	SMAL FROM	Intended recipient

HELO – This command is used by the client to identify itself. The argument is the domain name of the client host. The format is **HELO: challenger.atc.fhda.edu**

MAIL FROM. This command is used by the client to identify the sender of the message. The argument is the e-mail address of the sender (local part plus the domain name). The format is **MAIL FROM: forouzan@challenger.atc.fhda.edu**

RCPT TO – This command is used by the client to identify the intended recipient of the message. The argument is the e-mail address of the recipient. If there are multiple recipients, the command is repeated. The format is **RCPT TO: betsy@mcgraw-hill.com**

DATA – This command is used to send the actual message. All lines that follow the DATA command are treated as the mail message. The message is terminated by a line containing just one period. The format is

```
DATA

This is the message
to be sent to the McGraw-Hill
Company.
.
```

QUIT – This command terminates the message. The format is **QUIT**

RSET – This command aborts the current mail transaction. The stored information about the sender and recipient is deleted. The connection will be reset. **RSET**

VRFY – This command is used to verify the address of the recipient, which is sent as the argument. The sender can ask the receiver to confirm that a name identifies a valid recipient. Its format is **VRFY: betsy@mcgraw-hill.com**

NOOP – This command is used by the client to check the status of the recipient. It requires an answer from the recipient. Its format is **NOOP**

TURN – This command lets the sender and the recipient switch positions, whereby the sender becomes the recipient and vice versa. However, most SMTP implementations today do not support this feature. The format is **TURN**

EXPN – This command asks the receiving host to expand the mailing list sent as the arguments and to return the mailbox addresses of the recipients that comprise the list. The format is **EXPN: x y z**

HELP – This command asks the recipient to send information about the command sent as the argument. The format is **HELP: mail**
SEND FROM – This command specifies that the mail is to be delivered to the terminal of the recipient, and not the mailbox. If the recipient is not logged in, the mail is bounced back. The argument is the address of the sender. The format is **SEND FROM: forouzan@fhda.atc.edu**

SMOL FROM – This command specifies that the mail is to be delivered to the terminal or the mailbox of the recipient. This means that if the recipient is logged in, the mail is delivered only to the terminal.
If the recipient is not logged in, the mail is delivered to the mailbox. The argument is the address of the sender. The format is **SMOL FROM: forouzan@fhda.atc.edu**

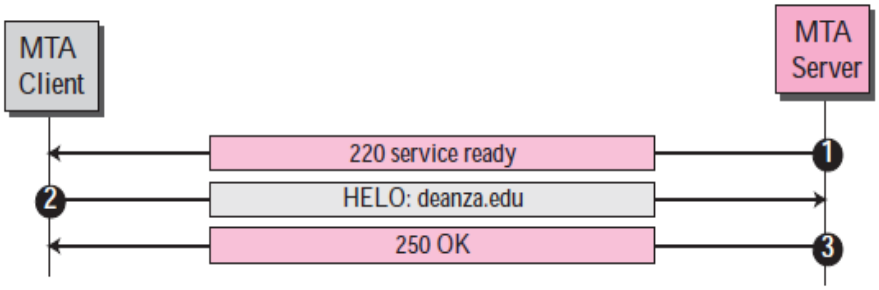
SMAL FROM. This command specifies that the mail is to be delivered to the terminal and the mailbox of the recipient. This means that if the recipient is logged in, the mail is delivered to the terminal and the mailbox.
If the recipient is not logged in, the mail is delivered only to the mailbox. The argument is the address of the sender. The format is **SMAL FROM: forouzan@fhda.atc.edu**

Responses –
Responses are sent from the server to the client. A response is a three-digit code that may be followed by additional textual information. Table – lists some of the responses.

Mail Transfer Phases

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

Connection Establishment –
After a client has made a TCP connection to the well-known port 25, the SMTP server starts the connection phase. This phase involves the following three steps, which are illustrated in Figure –

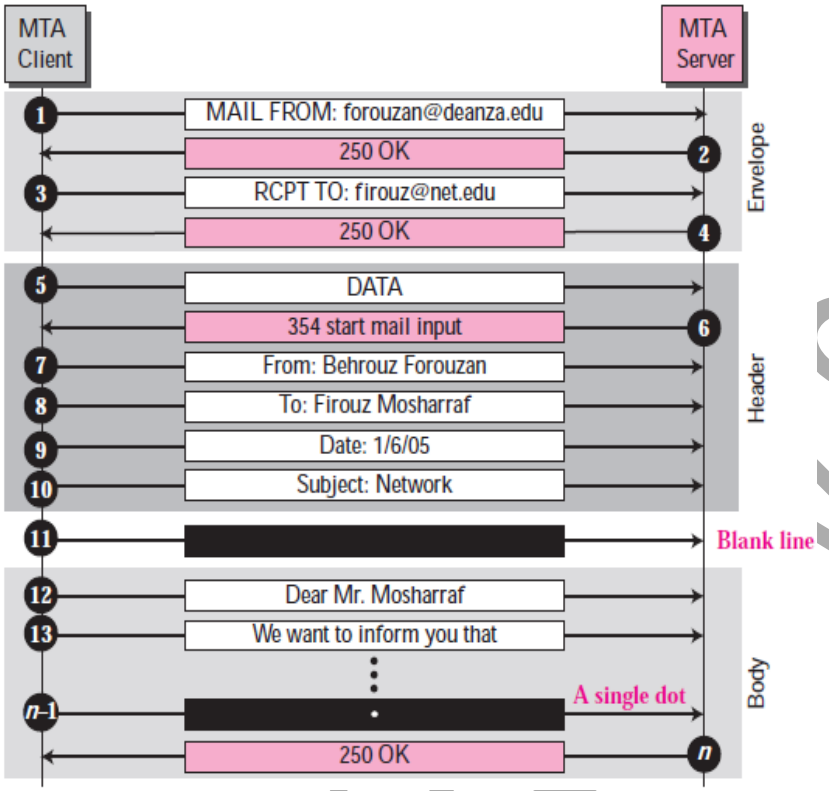


1. The server sends code 220 (service ready) to tell the client that it is ready to receive mail. If the server is not ready, it sends code 421 (service not available).
2. The client sends the HELO message to identify itself using its domain name address. This step is necessary to inform the server of the domain name of the client. Remember that during TCP connection establishment, the sender and receiver know each other through their IP addresses.
3. The server responds with code 250 (request command completed) or some other code depending on the situation.

Message Transfer –

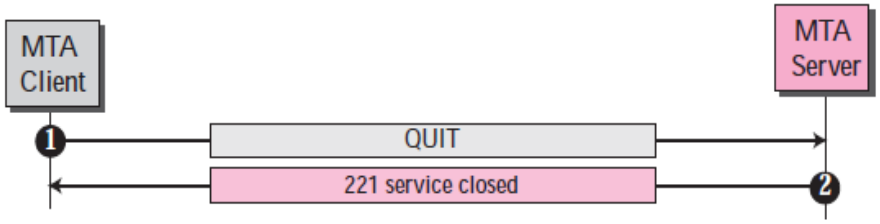
After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps. Steps 3 and 4 are repeated if there is more than one recipient (see Figure).

1. The client sends the MAIL FROM message to introduce the sender of the message. It includes the mail address of the sender (mailbox and the domain name). This step is needed to give the server the return mail address for returning errors and reporting messages.
2. The server responds with code 250 or some other appropriate code.
3. The client sends the RCPT TO (recipient) message, which includes the mail address of the recipient.
4. The server responds with code 250 or some other appropriate code.
5. The client sends the DATA message to initialize the message transfer.
6. The server responds with code 354 (start mail input) or some other appropriate message.
7. The client sends the contents of the message in consecutive lines. Each line is terminated by a two-character end-of-line token (carriage return and line feed). The message is terminated by a line containing just one period.
8. The server responds with code 250 (OK) or some other appropriate code.



***Connection Termination* –**

After the message is transferred successfully, the client terminates the connection. This phase involves two steps (see Figure).



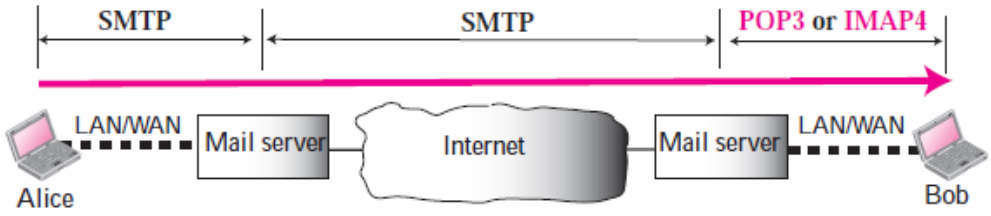
1. The client sends the QUIT command.
2. The server responds with code 221 or some other appropriate code.

After the connection termination phase, the TCP connection must be closed.

MESSAGE ACCESS AGENT: POP AND IMAP –

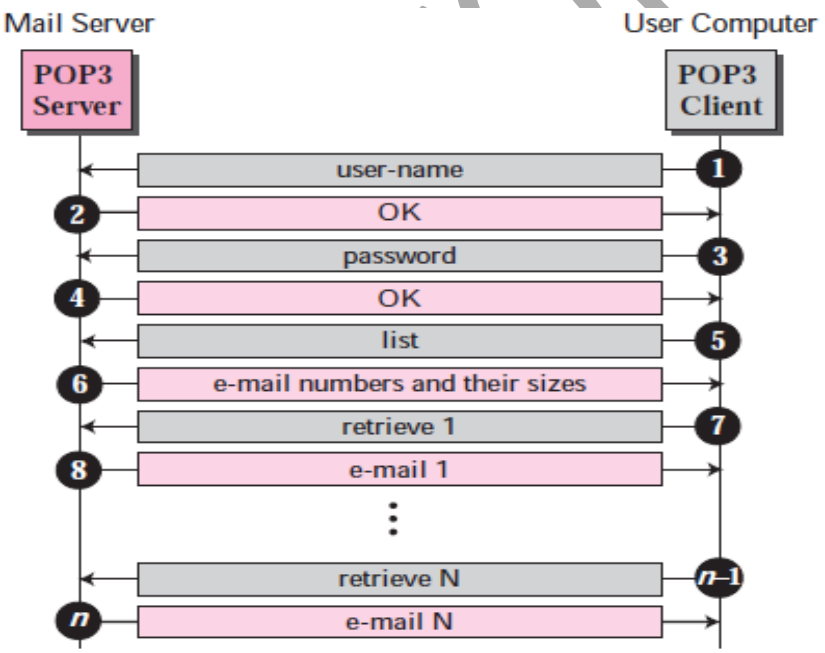
- The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from the client to the server.
- In other words, the direction of the bulk data (messages) is from the client to the server. On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data are from the server to the client. The third stage uses a message access agent.

- Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4). Figure – shows the position of these two protocols in the most common situation (fourth scenario).



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 its e-mail from the mailbox on the mail server.
- The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. Figure – shows an example of downloading using POP3.



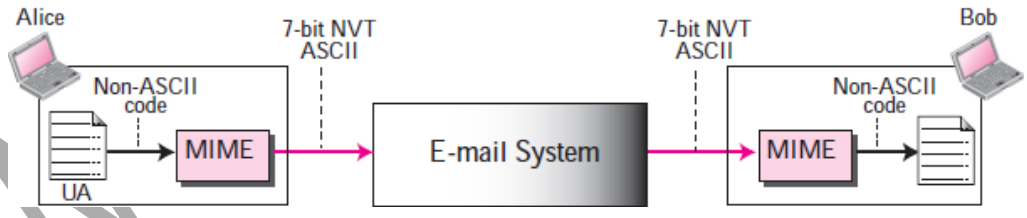
- POP3 has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval.
- The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying.
- The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

IMAP4 –

- Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4).** IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.
- POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.
- IMAP4 provides the following extra functions:
  1. A user can check the e-mail header prior to downloading.
  2. A user can search the contents of the e-mail for a specific string of characters prior to downloading.
  3. A user can partially download e-mail. This is especially useful if bandwidth is limited and the email contains multimedia with high bandwidth requirements.
  4.A user can create, delete, or rename mailboxes on the mail server.
  5. A user can create a hierarchy of mailboxes in a folder for e-mail storage.

MIME –

- Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.
- **Multipurpose Internet Mail Extensions (MIME)** is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving site is transformed back to the original data.
- We can think of MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa, as shown in Figure –

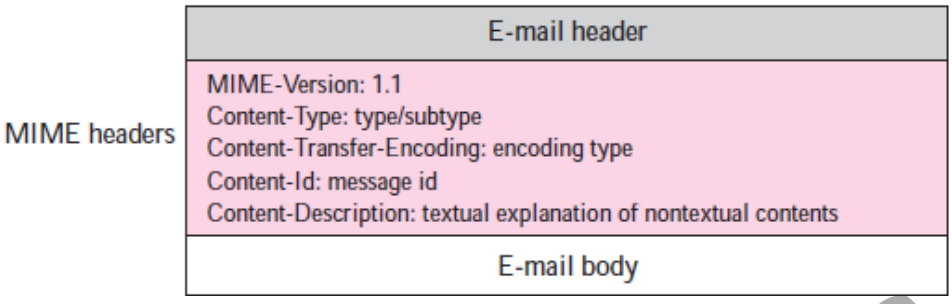


MIME Headers –

MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:

1. MIME-Version
2. Content-Type
3. Content-Transfer-Encoding
4. Content-Id
5. Content-Description

Figure – shows the MIME headers.



MIME-Version –
This header defines the version of MIME used. The current version is 1.1.
Content-Type –
This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters. MIME allows seven different types of data, listed in Table –

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Appendix E)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, 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 (eight-bit bytes)

Content-Transfer-Encoding –
This header defines the method used to encode the messages into 0s and 1s for transport: **Content-Transfer-Encoding: <type>** The five types of encoding methods are listed in Table –

Type	*Description*
7bit	NVT ASCII characters and short lines
8bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data are encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters are encoded as an equal sign plus an ASCII code

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

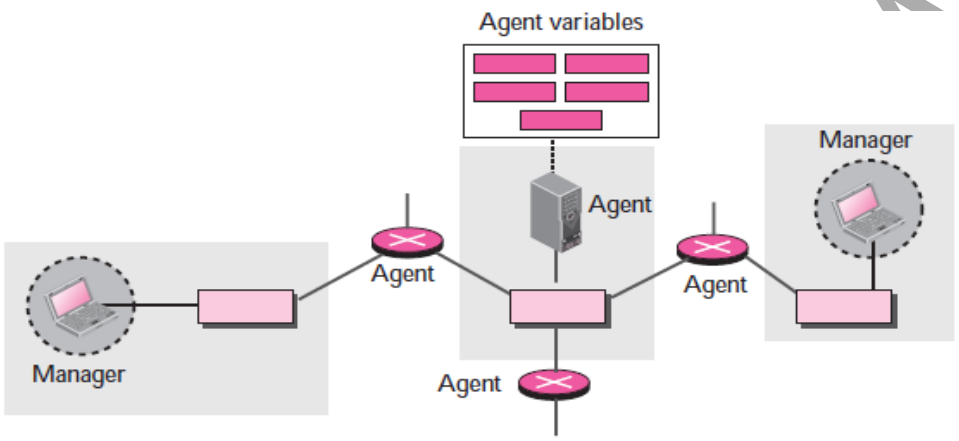
This header defines whether the body is image, audio, or video.

Network Management: SNMP

The **Simple Network Management Protocol (SNMP)** is a framework for managing devices in an internet using the TCP/IP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet.

CONCEPT –

SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers (see Figure).



- SNMP is an application-level protocol in which a few manager stations control a set of agents. The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks.
- In other words, SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology. It can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers.

Managers and Agents

- A management station, called a **manager,** is a host that runs the SNMP client program. A managed station, called an **agent,** is a router (or a host) that runs the SNMP server program. Management is achieved through simple interaction between a manager and an agent.
- The agent keeps performance information in a database. The manager has access to the values in the database. For example, a router can store in appropriate variables the number of packets received and forwarded. The manager can fetch and compare the values of these two variables to see if the router is congested or not.
- The manager can also make the router perform certain actions. For example, a router periodically checks the value of a reboot counter to see when it should reboot itself. It reboots itself, for example, if the value of the counter is 0.

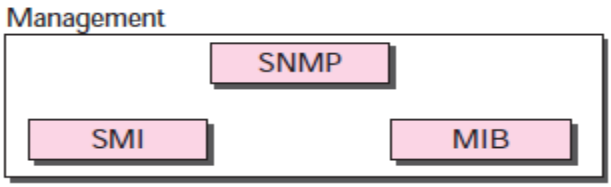
- The manager can use this feature to reboot the agent remotely at any time. It simply sends a packet to force a 0 value in the counter.
- Agents can also contribute to the management process. The server program running on the agent can check the environment and, if it notices something unusual, it can send a warning message (called a **trap**) to the manager.

In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by warning the manager of an unusual situation.

MANAGEMENT COMPONENTS –

To do management tasks, SNMP uses two other protocols: **Structure of Management Information (SMI)** and **Management Information Base (MIB).** In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB, as shown in Figure –



Role of SNMP –

- SNMP has some very specific roles in network management. It defines the format of the packet to be sent from a manager to an agent and vice versa. It also interprets the result and creates statistics (often with the help of other management software).
- The packets exchanged contain the object (variable) names and their status (values). SNMP is responsible for reading and changing these values.

Role of SMI –

- To use SNMP, we need rules. We need rules for naming objects. This is particularly important because the objects in SNMP form a hierarchical structure (an object may have a parent object and some child objects). Part of a name can be inherited from the parent.
- We also need rules to define the type of the objects. What types of objects are handled by SNMP? Can SNMP handle simple types or structured types? How many simple types are available? What are the sizes of these types? What is the range of these types? In addition, how are each of these types encoded?
- SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type. SMI is a collection of general rules to name objects and to list their types. The association of an object with the type is not done by SMI.

Role of MIB –

- We hope it is clear that we need another protocol. For each entity to be managed, this protocol must define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object.
- This protocol is MIB. MIB creates a set of objects defined for each entity similar to a database (mostly meta data in a database, names and types without values).

An Analogy –

The three network management components are similar to what we need when we write a program in a computer language to solve a problem. Figure – shows the analogy.

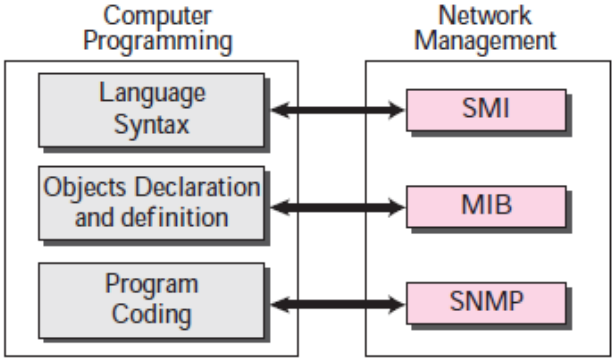


Figure *Comparing computer programming and network management*

SMI –

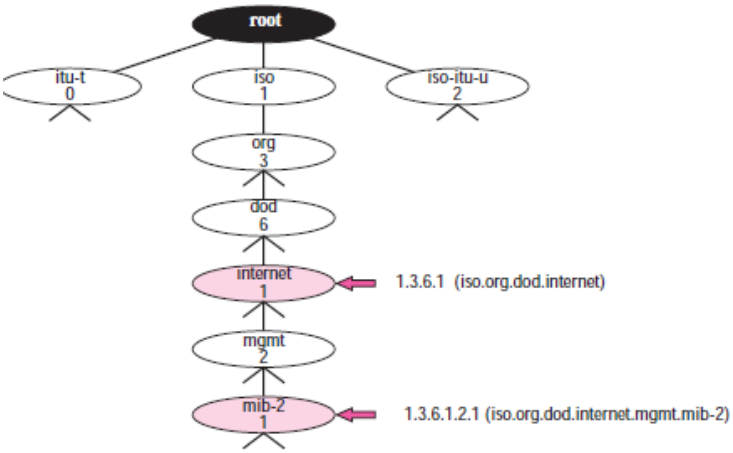
The Structure of Management Information, version 2 (SMIv2) is a component for network management. Its functions are:

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, etc.) have a unique name. To name objects globally, SMI uses an **object identifier,** which is a hierarchical identifier based on a tree structure (see Figure). Each object can be defined using a sequence of integers separated by dots.



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.1)** definitions and adds some new definitions. In other words, SMI is both a subset and a superset of ASN.1.
SMI has two broad categories of data type: *simple* and *structured*. We first define the simple types and then show how the structured types can be constructed from the simple ones.

Simple Type –

The **simple data types** are atomic data types. Some of them are taken directly from ASN.1; some are added by SMI. The most important ones are given in – The first five are from ASN.1; the next seven are defined 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 zero to 2^{32}; when it reaches its maximum value it wraps back to zero
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 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

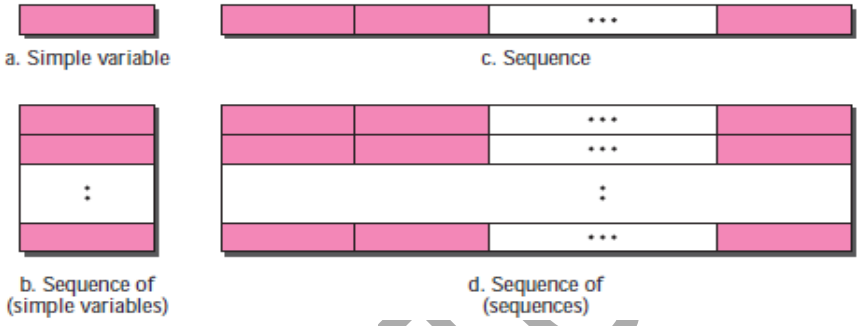
Structured Type –

By combining simple and structured data types, we can make new structured data types. SMI defines two **structured data types:** *sequence* and *sequence of*.

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

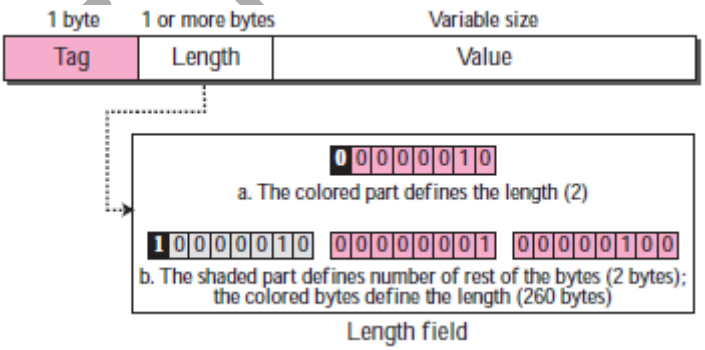
Sequence of.A *sequence of* data type is a combination of simple data types all of the same type or a combination of sequence data types all of the same type. It is analogous to the concept of an *array* used in programming languages such as C.

Figure – shows a conceptual view of data types.



Encoding Method –

SMI uses another standard, **Basic Encoding Rules (BER),** to encode data to be transmitted over the network. BER specifies that each piece of data be encoded in triplet format: tag, length, and value, as illustrated in Figure –

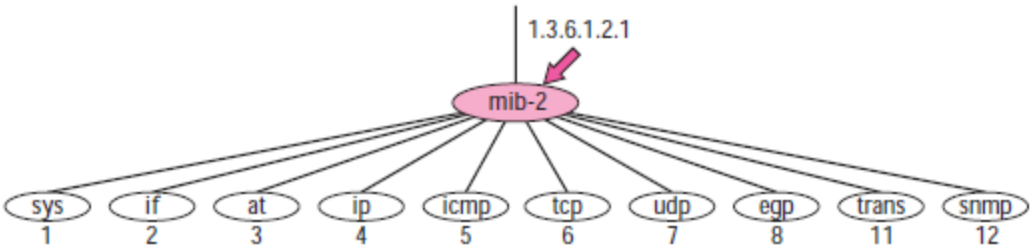


MIB

The Management Information Base, version 2 (MIB2) is the second component used in network management. Each agent has its own MIB2, which is a collection of all the objects that the manager can manage. The objects in MIB2 are categorized under

10 different groups: system, interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp.

These groups are under the mib-2 object in the object identifier tree (see Figure). Each group has defined variables and/or tables.



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 (*interface*) defines information about all of 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.

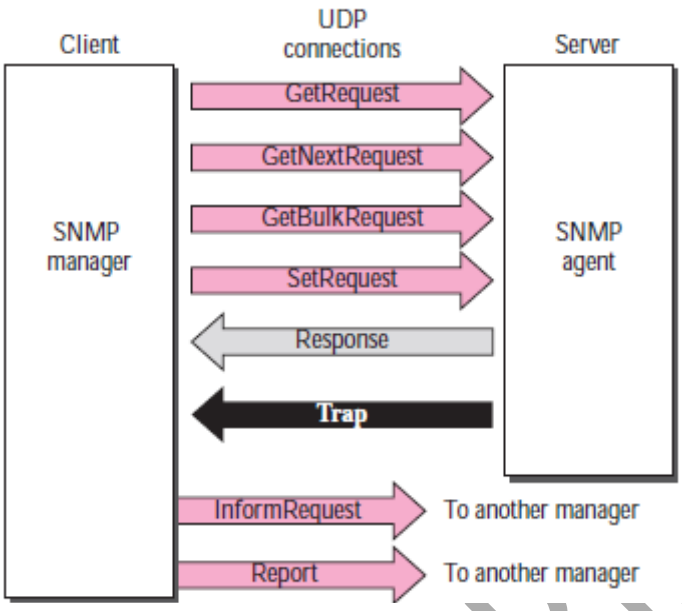
SNMP –

SNMP uses both SMI and MIB in Internet network management. It is an application program that allows:

1. **1.** A manager to retrieve the value of an object defined in an agent.
2. **2.** A manager to store a value in an object defined in an agent.
3. **3.** An agent to send an alarm message about an abnormal situation to the manager.

PDUs –

SNMPv3 defines eight types of protocol data units (or PDUs): GetRequest, GetNext-Request, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report (see Figure).



GetRequest

The GetRequest PDU is sent from the manager (client) to the agent (server) to retrieve the value of a variable or a set of variables.

GetNextRequest

The GetNextRequest PDU is sent from the manager to the agent to retrieve the value of a variable. The retrieved value is the value of the object following the defined ObjectId in the PDU. It is mostly used to retrieve the values of the entries in a table. If the manager does not know the indexes of the entries, it cannot retrieve the values. However, it can use GetNextRequest and define the ObjectId of the table. Because the first entry has the ObjectId immediately after the ObjectId of the table, the value of the first entry is returned. The manager can use this ObjectId to get the value of the next one, and so on.

GetBulkRequest

The GetBulkRequest PDU is sent from the manager to the agent to retrieve a large amount of data. It can be used instead of multiple GetRequest and GetNextRequest PDUs.

SetRequest

The SetRequest PDU is sent from the manager to the agent to set (store) a value in a variable.

Response

The Response PDU is sent from an agent to a manager in response to GetRequest or GetNextRequest. It contains the value(s) of the variable(s) requested by the manager.

Trap

The **Trap** (also called SNMPv2 Trap to distinguish it from SNMPv1 Trap) PDU is sent from the agent to the manager to report an event. For example, if the agent is rebooted, it informs the manager and reports the time of rebooting.

InformRequest

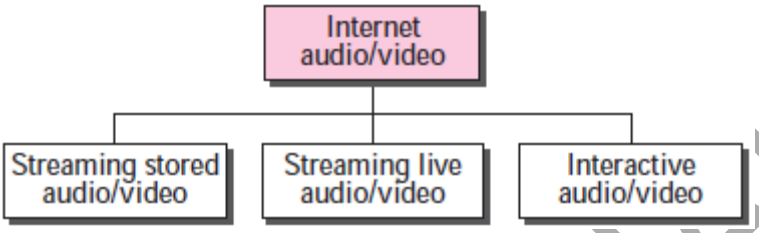
The InformRequest PDU is sent from one manager to another remote manager to get the value of some variables from agents under the control of the remote manager. The remote manager responds with a Response PDU.

Report

The Report PDU is designed to report some types of errors between managers. It is not yet in use.

Multimedia

We can divide audio and video services into three broad categories: **streaming stored audio/video, streaming live audio/video,** and **interactive audio/video,** as shown in Figure – Streaming means a user can listen (or watch) the file after the downloading has started.



- In the first category, streaming stored audio/video, the files are compressed and stored on a server. A client downloads the files through the Internet. This is sometimes referred to as **on-demand audio/video.** Examples of stored audio files are songs, symphonies, books on tape, and famous lectures. Examples of stored video files are movies, TV shows, and music video clips.
- In the second category, streaming live audio/video, a user listens to broadcast audio and video through the Internet. A good example of this type of application is the Internet radio. Some radio stations broadcast their programs only on the Internet; many broadcast them both on the Internet and on the air. Internet TV is not popular yet, but many people believe that TV stations will broadcast their programs on the Internet in the future.
- In the third category, interactive audio/video, people use the Internet to interactively communicate with one another. A good example of this application is Internet telephony and Internet teleconferencing.

DIGITIZING AUDIO AND VIDEO –

Digitizing Audio

- When sound is fed into a microphone, an electronic analog signal is generated that represents the sound amplitude as a function of time. The signal is called an *analog audio signal*. An analog signal, such as audio, can be digitized to produce a digital signal.
- According to the Nyquist theorem, if the highest frequency of the signal is *f*, we need to sample the signal 2*f* times per second. There are other methods for digitizing an audio signal, but the principle is the same.
- Voice is sampled at 8,000 samples per second with 8 bits per sample. This results in a digital signal of 64 kbps. Music is sampled at 44,100 samples per second with 16 bits per sample. This results in a digital signal of 705.6 kbps for monaural and 1.411 Mbps for stereo.

Digitizing Video –

- A video consists of a sequence of frames. If the frames are displayed on the screen fast enough, we get an impression of motion. The reason is that our eyes cannot distinguish

the rapidly flashing frames as individual ones. There is no standard number of frames per second; in North America 25 frames per second is common.

- However, to avoid a condition known as flickering, a frame needs to be refreshed. The TV industry repaints each frame twice. This means 50 frames need to be sent, or if there is memory at the sender site, 25 frames with each frame repainted from the memory.
- Each frame is divided into small grids, called picture elements or **pixels.** For blackand – white TV, each 8-bit pixel represents one of 256 different gray levels. For a color TV, each pixel is 24 bits, with 8 bits for each primary color (red, green, and blue).
- We can calculate the number of bits in a second for a specific resolution. In the lowest resolution a color frame is made of 1,024 × 768 pixels. This means that we need

$$2 \times 25 \times 1{,}024 \times 768 \times 24 = 944 \text{ Mbps}$$

- This data rate needs a very high data rate technology such as SONET. To send video using lower-rate technologies, we need to compress the video.

AUDIO AND VIDEO COMPRESSION

To send audio or video over the Internet requires **compression.**

Audio Compression –

Audio compression can be used for speech or music. For speech, we need to compress a 64-kHz digitized signal; for music, we need to compress a 1.411-MHz signal. Two categories of techniques are used for audio compression: predictive encoding and perceptual encoding.

Predictive Encoding –

In **predictive encoding,** the differences between the samples are encoded instead of encoding all the sampled values. This type of compression is normally used for speech.

Several standards have been defined such as GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 or 5.3 kbps).

Perceptual Encoding: MP3 –

The most common compression technique that is used to create CD-quality audio is based on the **perceptual encoding** technique. As we mentioned before, this type of audio needs at least 1.411 Mbps; this cannot be sent over the Internet without compression. **MP3** (MPEG audio layer 3), a part of the MPEG standard (discussed in the video compression section), uses this technique.

Perceptual encoding is based on the science of psychoacoustics, which is the study of how people perceive sound. The idea is based on some flaws in our auditory system:

Some sounds can mask other sounds. Masking can happen in frequency and time.

In **frequency masking,** a loud sound in a frequency range can partially or totally mask a so after sound in another frequency range. For example, we cannot hear what our dance partner says in a room where a loud heavy metal band is performing.

In **temporal masking,** a loud sound can numb our ears for a short time even after the sound has stopped.

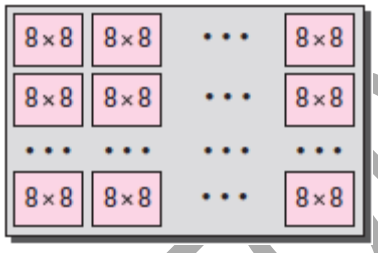
MP3 uses these two phenomena, frequency and temporal masking, to compress audio signals. The technique analyzes and divides the spectrum into several groups.

Zero bits are allocated to the frequency ranges that are totally masked. A small number of bits are allocated to the frequency ranges that are partially masked. A larger number of bits are allocated to the frequency ranges that are not masked.

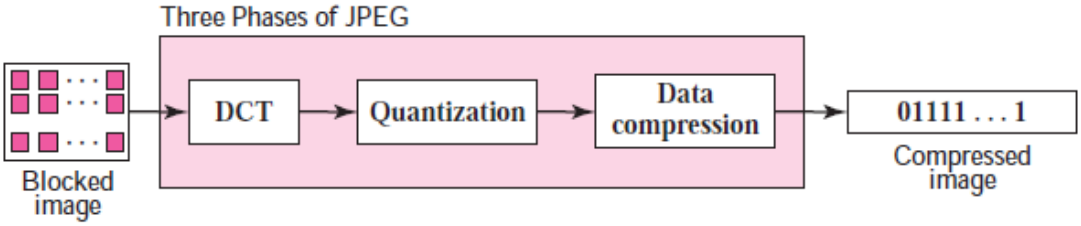
MP3 produces three data rates: 96 kbps, 128 kbps, and 160 kbps. The rate is based on the range of the frequencies in the original analog audio.

Video Compression –
video is composed of multiple frames. Each frame is one image. We can compress video by first compressing images. Two standards are prevalent in the market. **Joint Photographic Experts Group (JPEG)** is used to compress images. **Moving Picture Experts Group (MPEG)** is used to compress video.

Image Compression: JPEG
if the picture is not in color (gray scale), each pixel can be represented by an 8-bit integer (256 levels). If the picture is in color, each pixel can be represented by 24 bits (3 × 8 bits), with each 8 bits representing red, blue, or green (RBG). To simplify the discussion, we concentrate on a gray scale picture. In JPEG, a gray scale picture is divided into blocks of 8 × 8 pixels (see Figure).



The purpose of dividing the picture into blocks is to decrease the number of calculations because, the number of mathematical operations for each picture is the square of the number of units.
The whole idea of JPEG is to change the picture into a linear (vector) set of numbers that reveals the redundancies. The redundancies (lack of changes) can then be removed by using one of the text compression methods. A simplified scheme of the process is shown in Figure –



Discrete Cosine Transform (DCT) In this step, each block of 64 pixels goes through a transformation called the **discrete cosine transform (DCT).** The transformation changes the 64 values so that the relative relationships between pixels are kept but the redundancies are revealed.

Quantization –

- After the *T* table is created, the values are quantized to reduce the number of bits needed for encoding. Previously in **quantization,** we dropped the fraction from each value and kept the integer part. Here, we divide the number by a constant and then drop the fraction.
- This reduces the required number of bits even more. In most implementations, a quantizing table (8 by 8) defines how to quantize each value. The divisor depends on the

position of the value in the *T* table. This is done to optimize the number of bits and the number of 0s for each particular application.

- Note that the only phase in the process that is not completely reversible is the quantizing phase. We lose some information here that is not recoverable. As a matter of fact, the only reason that JPEG is called *lossy compression* is because of this quantization phase.

Compression –

- After quantization, the values are read from the table, and redundant 0s are removed. However, to cluster the 0s together, the table is read diagonally in a zigzag fashion rather than row by row or column by column.
- The reason is that if the picture changes smoothly, the bottom right corner of the *T* table is all 0s.

Video Compression: MPEG

The Moving Picture Experts Group (MPEG) method is used to compress video. In principle, a motion picture is a rapid flow of a set of frames, where each frame is an image. In other words, a frame is a spatial combination of pixels, and a video is a temporal combination of frames that are sent one after another. Compressing video, then, means spatially compressing each frame and temporally compressing a set of frames.

Spatial Compression The **spatial compression** of each frame is done with JPEG (or a modification of it). Each frame is a picture that can be independently compressed.

Temporal Compression In **temporal compression,** redundant frames are removed. When we watch television, we receive 50 frames per second. However, most of the consecutive frames are almost the same. For example, when someone is talking, most of the frame is the same as the previous one except for the segment of the frame around the lips, which changes from one frame to another.

To temporally compress data, the MPEG method first divides frames into three categories: I-frames, P-frames, and B-frames. Figure 25.8 shows a sample sequence of frames.

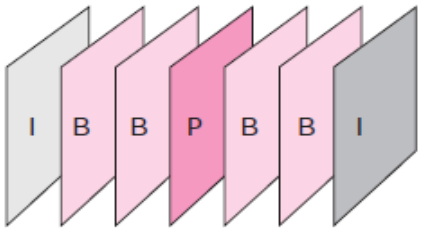
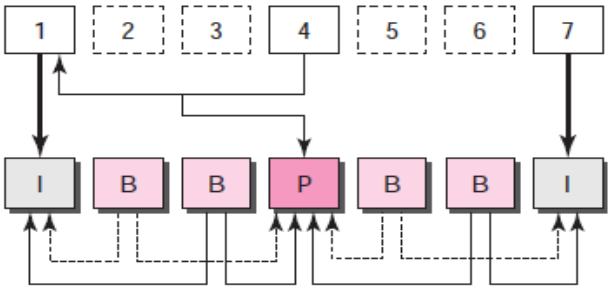


Figure shows how I-, P-, and B-frames are constructed from a series of seven frames.



- **I-frames.** An **intracoded frame (I-frame)** is an independent frame that is not related to any other frame (not to the frame sent before or to the frame sent after). They are present at regular intervals (e.g., every ninth frame is an I-frame).
- An I-frame must appear periodically to handle some sudden change in the frame that the previous and following frames cannot show. Also, when a video is broadcast, a viewer may tune at any time. If there is only one I-frame at the beginning of the broadcast, the viewer who tunes in late will not receive a complete picture.
- I-frames are independent of other frames and cannot be constructed from other frames.
- **P-frames.** A **predicted frame (P-frame)** is related to the preceding I-frame or P-frame. In other words, each P-frame contains only the changes from the preceding frame.
- The changes, however, cannot cover a big segment. For example, for a fast-moving object, the new changes may not be recorded in a P-frame. P-frames can be constructed only from previous I- or P-frames. P-frames carry much less information than other frame types and carry even fewer bits after compression.
- **B-frames.** A **bidirectional frame (B-frame)** is related to the preceding and following I-frame or P-frame. In other words, each B-frame is relative to the past and the future. Note that a B-frame is never related to another B-frame.

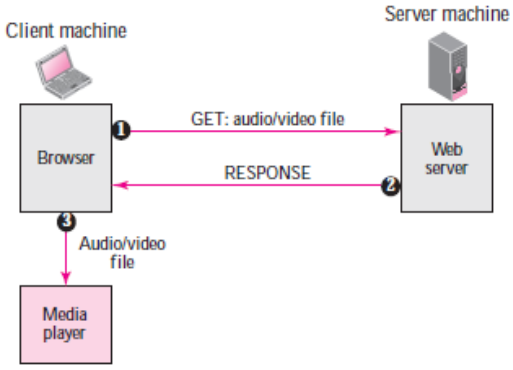
MPEG has gone through two versions. MPEG1 was designed for a CD-ROM with a data rate of 1.5 Mbps. MPEG2 was designed for high-quality DVD with a data rate of 3 to 6 Mbps.

STREAMING STORED AUDIO/VIDEO

The first is streaming stored audio and video. Downloading these types of files from a Web server can be different from downloading other types of files. To understand the concept, let us discuss three approaches, each with a different complexity.

First Approach: Using a Web Server

- A compressed audio/video file can be downloaded as a text file. The client (browser) can use the services of HTTP and send a GET message to download the file. The Web server can send the compressed file to the browser. The browser can then use a help application, normally called a **media player,** to play the file. Figure – shows this approach.

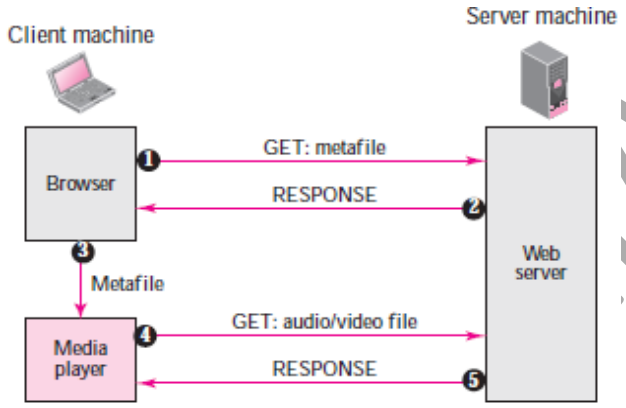


- This approach is very simple and does not involve *streaming*. However, it has a drawback. An audio/video file is usually large even after compression. An audio file may contain tens of megabits, and a video file may contain hundreds of megabits.

- In this approach, the file needs to download completely before it can be played. Using contemporary data rates, the user needs some seconds or tens of seconds before the file can be played.

Second Approach: Using a Web Server with Metafile

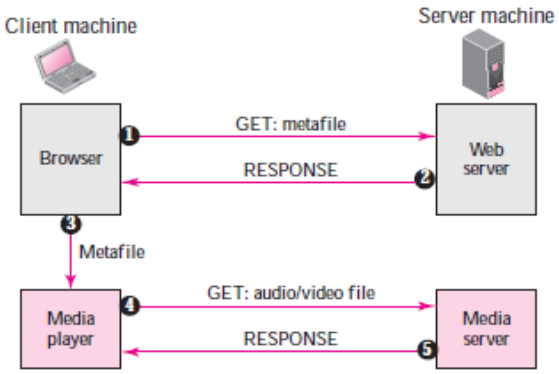
- In another approach, the media player is directly connected to the Web server for downloading the audio/video file. The Web server stores two files: the actual audio/video file and a **metafile** that holds information about the audio/video file. Figure – shows the steps in this approach.



1. The HTTP client accesses the Web server using the GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the audio/video file.
5. The Web server responds.

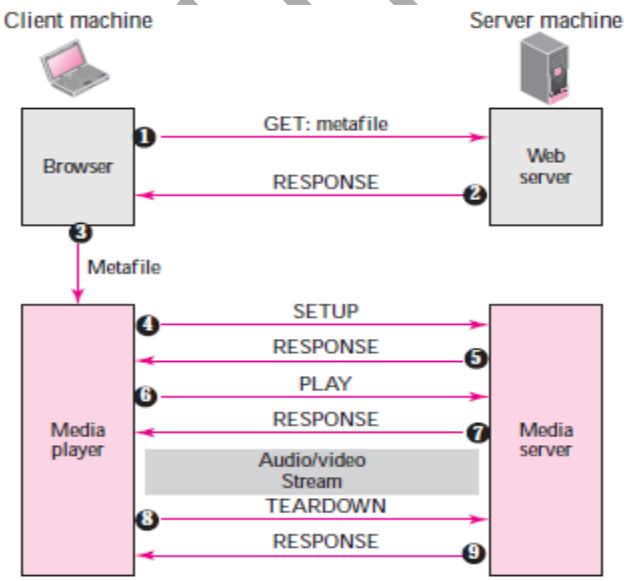
Third Approach: Using a Media Server

- The problem with the second approach is that the browser and the media player both use the services of HTTP. HTTP is designed to run over TCP. This is appropriate for retrieving the metafile, but not for retrieving the audio/video file.
- The reason is that TCP retransmits a lost or damaged segment, which is counter to the philosophy of streaming.
- We need to dismiss TCP and its error control; we need to use UDP. However, HTTP, which accesses the Web server, and the Web server itself are designed for TCP; we need another server, a **media server.** Figure – shows the concept.



1. The HTTP client accesses the Web server using a GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the media server to download the file. Downloading can take place by any protocol that uses UDP.
5. The media server responds.

Fourth Approach: Using a Media Server and RTSP –
The **Real-Time Streaming Protocol (RTSP)** is a control protocol designed to add more functionalities to the streaming process. Using RTSP, we can control the playing of audio/video. RTSP is an out-of-band control protocol that is similar to the second connection in FTP. Figure – shows a media server and RTSP.



1. The HTTP client accesses the Web server using a GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player sends a SETUP message to create a connection with the media server.
5. The media server responds.
6. The media player sends a PLAY message to start playing (downloading).

7. The audio/video file is downloaded using another protocol that runs over UDP.
8. The connection is broken using the TEARDOWN message.
9. The media server responds.

The media player can send other types of messages. For example, a PAUSE message temporarily stops the downloading; downloading can be resumed with a PLAY message.

25.6 REAL-TIME INTERACTIVE AUDIO/VIDEO

In real-time interactive audio/video, people communicate with one another in real time. The Internet phone or voice over IP is an example of this type of application. Video conferencing is another example that allows people to communicate visually and orally.

Characteristics

Some characteristics of real-time audio/video communication.

Time Relationship

Real-time data on a packet-switched network require the preservation of the time relationship between packets of a session.

Timestamp

One solution to jitter is the use of a **timestamp.** If each packet has a timestamp that shows the time it was produced relative to the first (or previous) packet, then the receiver can add this time to the time at which it starts the playback.

Playback Buffer –

To be able to separate the arrival time from the playback time, we need a buffer to store the data until they are played back. The buffer is referred to as a **playback buffer.** When a session begins (the first bit of the first packet arrives), the receiver delays playing the data until a threshold is reached.

Ordering –

In addition to time relationship information and timestamps for real-time traffic, one more feature is needed. We need a *sequence number* for each packet. The timestamp alone cannot inform the receiver if a packet is lost. The receiver has no way of knowing that the second packet has actually been lost. A sequence number to order the packets is needed to handle this situation.

Multicasting –

Multimedia play a primary role in audio and video conferencing. The traffic can be heavy, and the data are distributed using **multicasting** methods. Conferencing requires two-way communication between receivers and senders.

Translation –

Sometimes real-time traffic needs **translation.** A translator is a computer that can change the format of a high-bandwidth video signal to a lower-quality narrow bandwidth signal.

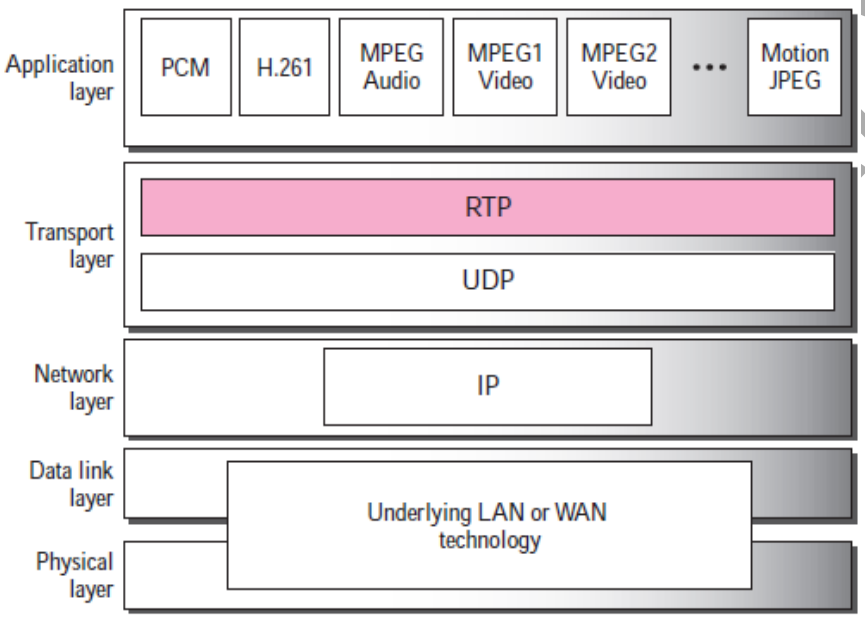
Mixing –

If there is more than one source that can send data at the same time (as in a video or audio conference), the traffic is made of multiple streams. To converge the traffic to one stream, data

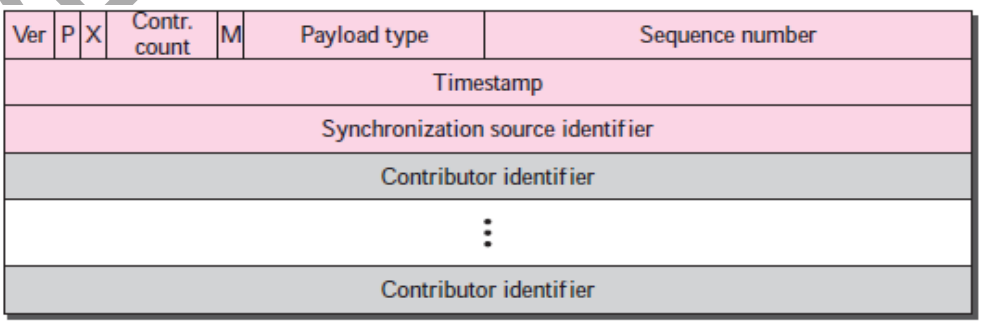
from different sources can be mixed. A **mixer** mathematically adds signals coming from different sources to create one single signal.

RTP

Real-time Transport Protocol (RTP) is the protocol designed to handle real-time traffic on the Internet. RTP does not have a delivery mechanism (multicasting, port numbers, and so on); it must be used with UDP. RTP stands between UDP and the application program. The main contributions of RTP are timestamping, sequencing, and mixing facilities. Figure – shows the position of RTP in the protocol suite.



RTP Packet Format –
Figure – shows the format of the RTP packet header. The format is very simple and general enough to cover all real-time applications. An application that needs more information adds it to the beginning of its payload. A description of each field follows.

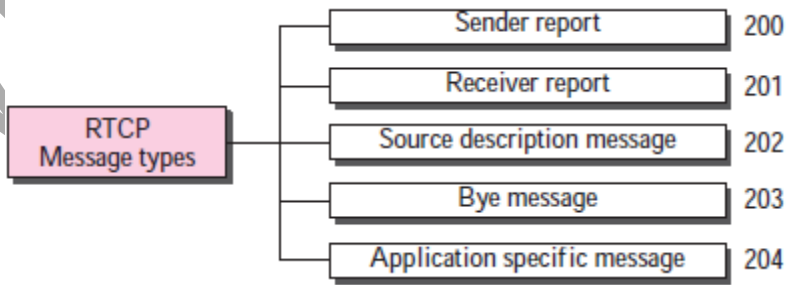


Ver – This 2-bit field defines the version number. The current version is 2.
P – This 1-bit field, if set to 1, indicates the presence of padding at the end of the packet. In this case, the value of the last byte in the padding defines the length of the padding. Padding is the norm if a packet is encrypted. There is no padding if the value of the P field is 0.

X – This 1-bit field, if set to 1, indicates an extra extension header between the basic header and the data. There is no extra extension header if the value of this field is 0.
Contributor count – This 4-bit field indicates the number of contributors. Note that we can have a maximum of 15 contributors because a 4-bit field only allows a number between 0 and 15.
M – This 1-bit field is a marker used by the application to indicate, for example, the end of its data.
Payload type – This 7-bit field indicates the type of the payload. Several payload types have been defined so far.
Sequence number – This field is 16 bits in length. It is used to number the RTP packets. The sequence number of the first packet is chosen randomly; it is incremented by 1 for each subsequent packet. The sequence number is used by the receiver to detect lost or out of order packets.
Timestamp – This is a 32-bit field that indicates the time relationship between packets. The timestamp for the first packet is a random number. For each succeeding packet, the value is the sum of the preceding timestamp plus the time the first byte is produced (sampled). The value of the clock tick depends on the application.
Synchronization source identifier – If there is only one source, this 32-bit field defines the source. However, if there are several sources, the mixer is the synchronization source and the other sources are contributors. The value of the source identifier is a random number chosen by the source. The protocol provides a strategy in case of conflict (two sources start with the same sequence number).
Contributor identifier – Each of these 32-bit identifiers (a maximum of 15) defines a source. When there is more than one source in a session, the mixer is the synchronization source and the remaining sources are the contributors.

RTCP

- RTP allows only one type of message, one that carries data from the source to the destination. In many cases, there is a need for other messages in a session. These messages control the flow and quality of data and allow the recipient to send feedback to the source or sources.
- **Real-Time Transport Control Protocol (RTCP)** is a protocol designed for this purpose. RTCP has five types of messages, as shown in Figure 25.20. The number next to each box defines the type of the message.



Sender Report

The sender report is sent periodically by the active senders in a conference to report transmission and reception statistics for all RTP packets sent during the interval. The sender report includes an absolute timestamp, which is the number of seconds elapsed
since midnight January 1, 1970. The absolute timestamp allows the receiver to synchronize different RTP messages. It is particularly important when both audio and video are transmitted (audio and video transmissions use separate relative timestamps).

Receiver Report

The receiver report is for passive participants, those that do not send RTP packets. The report informs the sender and other receivers about the quality of service.

Source Description Message

The source periodically sends a source description message to give additional information about itself. This information can be the name, e-mail address, telephone number, and address of the owner or controller of the source.

Bye Message

A source sends a bye message to shut down a stream. It allows the source to announce that it is leaving the conference. Although other sources can detect the absence of a source, this message is a direct announcement. It is also very useful to a mixer.

Application-Specific Message

The application-specific message is a packet for an application that wants to use new applications (not defined in the standard). It allows the definition of a new message type.