Unit 1

Chapter 1

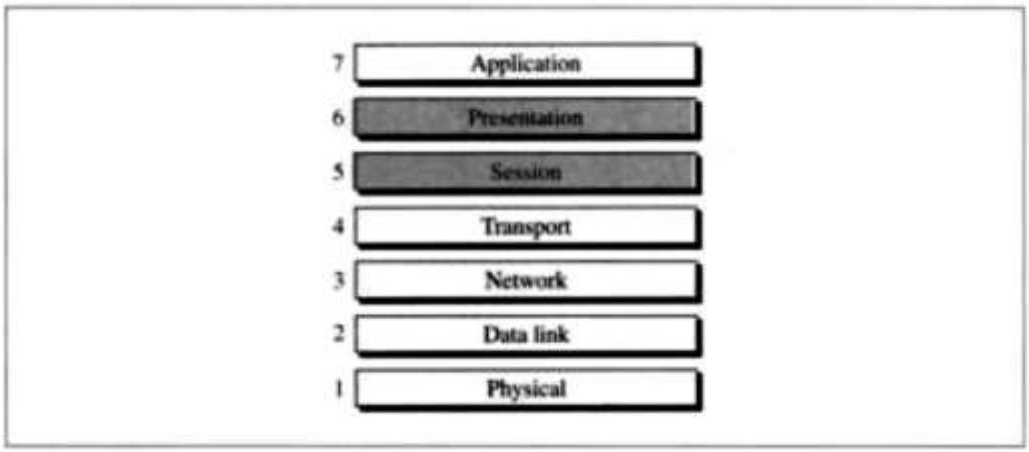
1.1 Introduction

Internet refers to network of networks. In this network each computer is recognized by a globally unique address known as IP address. A special computer DNS (Domain Name Server) is used to give name to the IP Address so that user can locate a computer by a name.

Internet uses basic protocols Like TCP and IP for transmission of data, protocols ARP and RARP are used for Address Resolution, protocols http and ftp are used for web designing and file uploading and downloading

1.2 OSI Model

Open Systems Interconnection, or OSI, model, was designed by the International Organization for Standardization (ISO). It is a seven layer model, OSI was never seriously implemented as a protocol stack, however: it is a theoretical model designed to show how a protocol stack should be implemented.



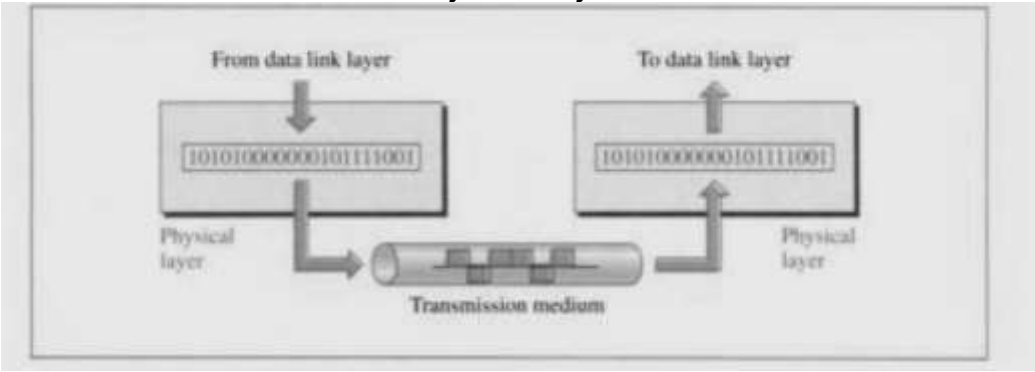
➢ Functions of Layers

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

➢ 1. Physical Layer

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

Physical Layer



The physical layer is responsible for transmitting individual bits from one node to the next.

The major duties of the physical layer are as follows:

■ **Physical characteristics of interfaces and media.:**

The physical layer defines the characteristics of the interface between the devices and the transmission media. It also defines the type of transmission medium

■ **Representation of bits:**

The physical layer data consists of a stream of bits (sequence of 0s or Is) without

any interpretation. To be transmitted, bits must be encoded into signals—electrical or optical. The physical layer defines the type of representation .how 0s and Is are changed to signals).

■ **Data rate**:

The **transmission rate**—the number of bits sent each second—is also defined by the physical layer. In other words, the physical layer defines the duration of a bit which is how long it lasts.

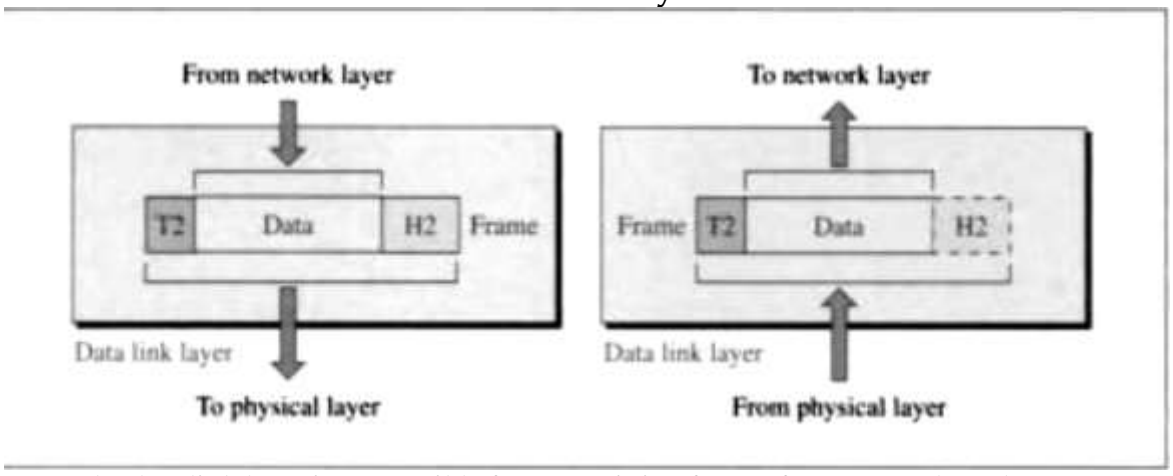
■ **Synchronization of bits**:

The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.

➢ 2. Data link Layer

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

Data Link Layer



The data link layer is responsible for transmitting frames from one node to the next.

The major duties of the data link layer are as follows

■ **Framing:**

The data link layer divides the stream of bits received from the network layer into manageable data units called **frames**

■ **Physical addressing:**

If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the connecting device that connects the network to the next one.

■ **Flow control:**

If the rate at which the data are absorbed by the receiver is less than the rate produced in the sender, the data link layer imposes a flow control mechanism to prevent overwhelming the receiver.

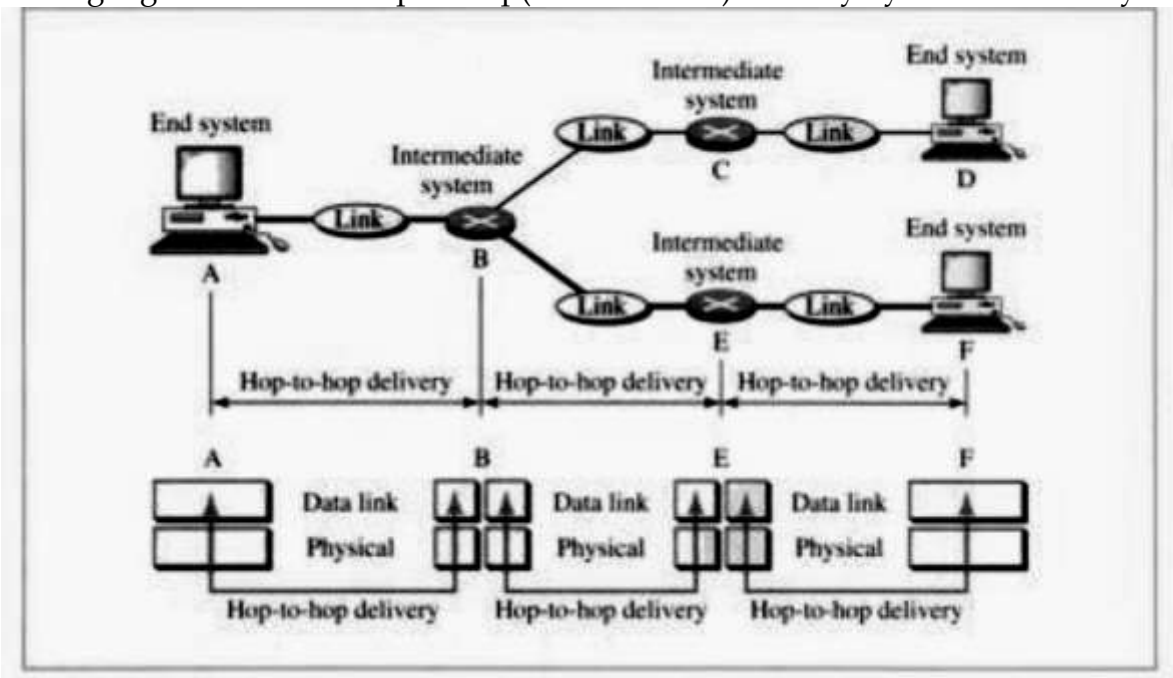
■ **Error control:**

The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to prevent duplication of frames. Error control is normally achieved through a trailer added to the end of the frame.

■ **Access control:**

When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

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

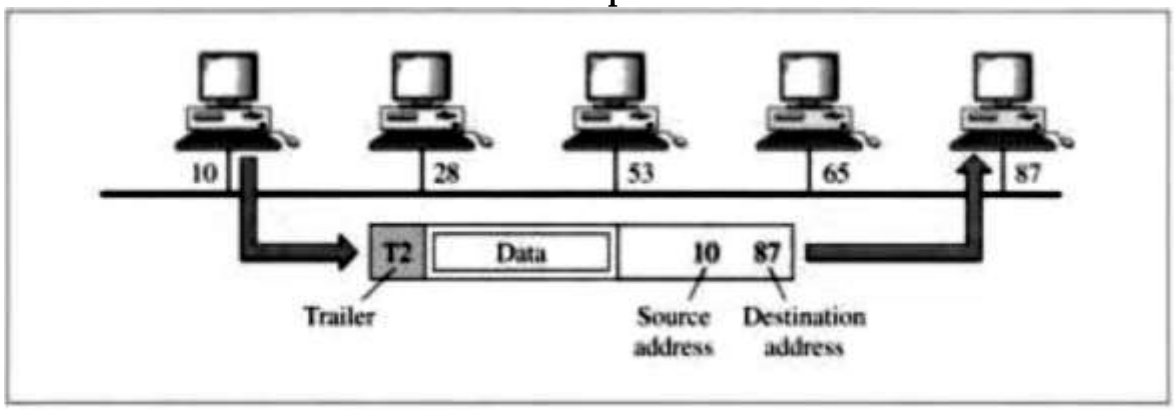


Example

In Figure 2.8 a node with physical address 10 sends a frame to a node with physical address 87. The two nodes are connected by a link. At the data link level this frame contains physical addresses in the header. These are the only addresses needed. The rest of the header contains other information needed at this level. The trailer usually contains extra

bits needed for error detection.

Example 1

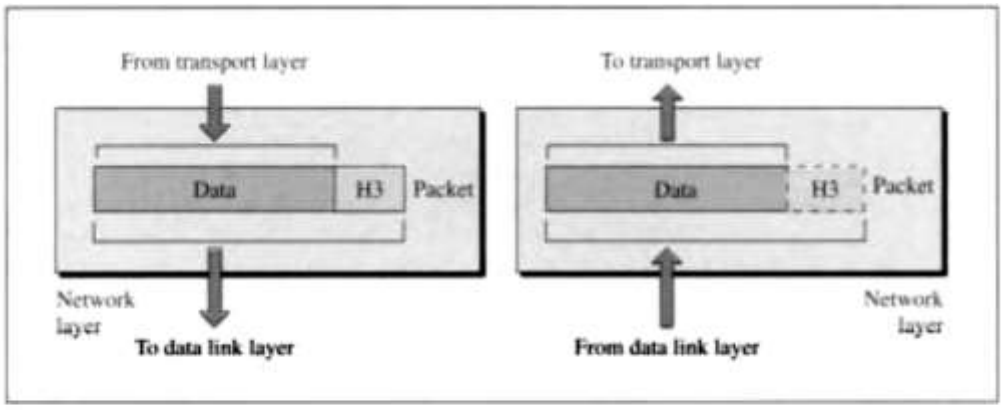


➢ Network Layer

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

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

Network Layer



The Network layer is responsible for the delivery of packets from the original source to the final destination.

The major duties of the network layer are as follows:

- **Logical addressing:**

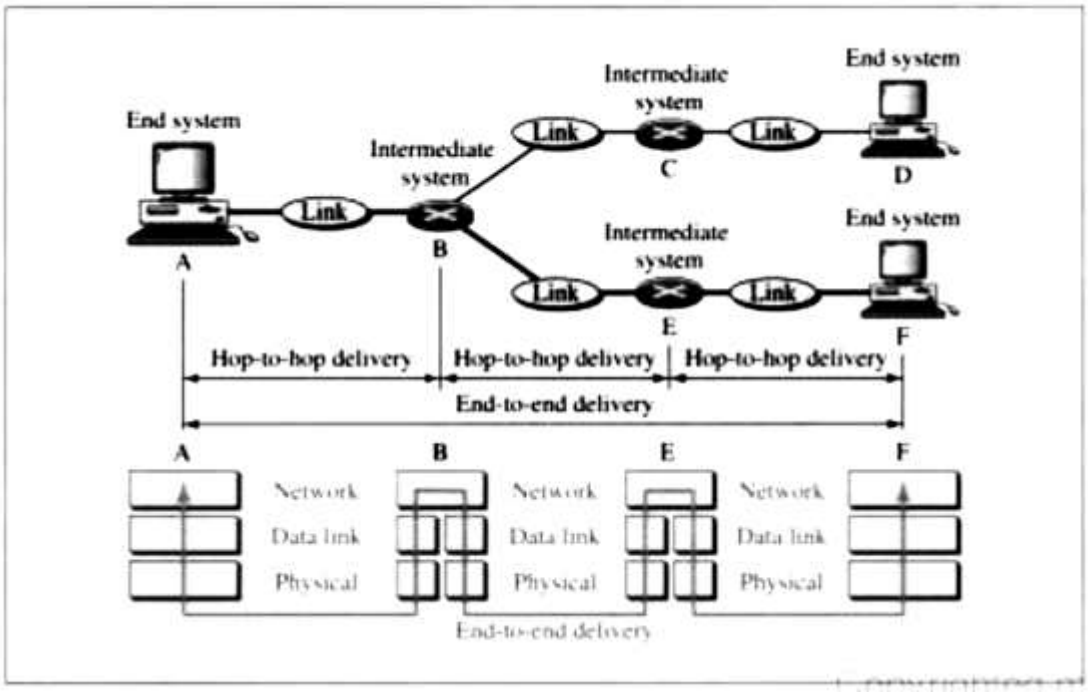
The physical addressing implemented by the data link layer handle the addressing problem locally. If a packet passes the network boundary. we need another addressing system to help distinguish the source and destination systems. The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver.

- **Routing:**

  When independent networks or links are connected to create an internet work (network of networks) or a large network, the connecting devices (called routers or switches) route or switch the packets to their final destination. One of the functions of the network layer is to provide this mechanism.

Following Figure illustrates source-to-destination delivery by the network layer.

Source-to-destination delivery

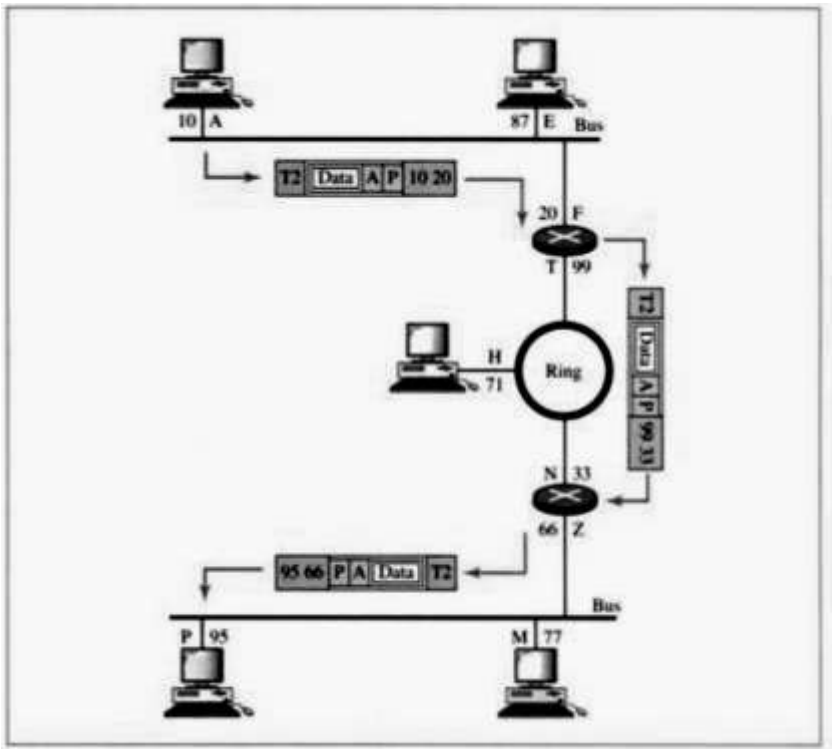


Example

In following figure of Example we want to send data from a node with network address A and physical address 10. Located on one LAN. To a node with a network address P and physical address 95 located on another LAN. Because the two devices are located on different network we cannot use physical addresses only, the physical addresses only have local jurisdiction. What we need here are universal addresses that

can pass through the LAN boundaries. The network
(logical) addresses have this characteristic. The packet at the network layer contains the logical addresses, which remain the same from the original source

Example

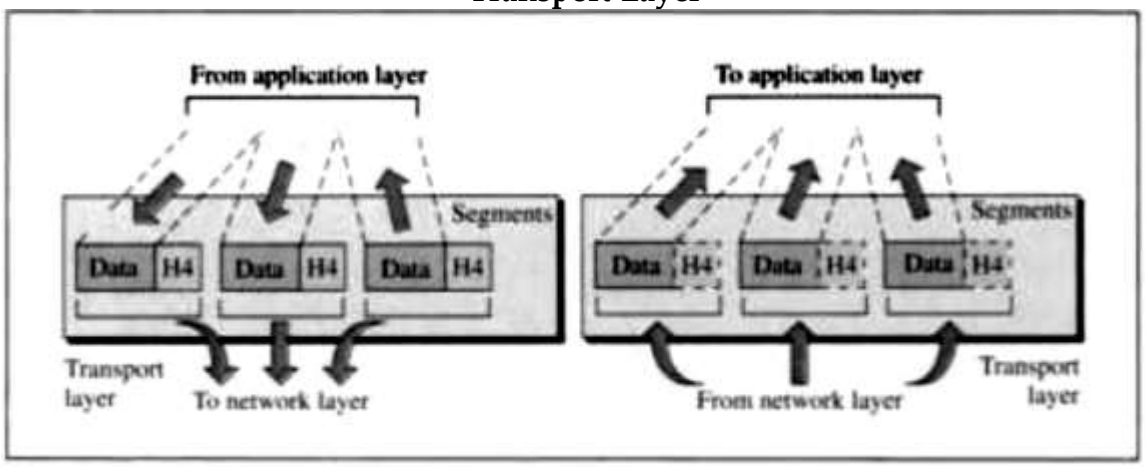


in the final destination (A and P. respectively, in the figure). They will not change when we go from network to network However, the physical addresses will change as the packet moves from one network lo another .The box with the R is a router (internet work device).

➢ Transport Layer

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

Transport Layer



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

The major duties of the transport layer are as follows:

■ **Port addressing**:

Computers often run several processes (running programs) at the same time. for this reason, process-to-process- delivery means delivery not only from one computer to the next but also from a specific process on one computer to a specific process on the other. The transport layer header must therefore include a type of address called **a port address**. The network layer gets each packet to the correct computer: the transport layer gets the entire message lo the correct process on that computer

- **Segmentation and reassembly:**

A message is divided into transmittable segments, each segment containing a sequence number .these numbers enable the transport layer to reassemble the message correctly upon arrival at the destination and to identify and replace packets that were lost in the transmission.

■ **Connection control:**

The transport layer can be either connectionless or connection-oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.

■ **Flow control:**

Like the data link layer, the transport layer is responsible for flow control.

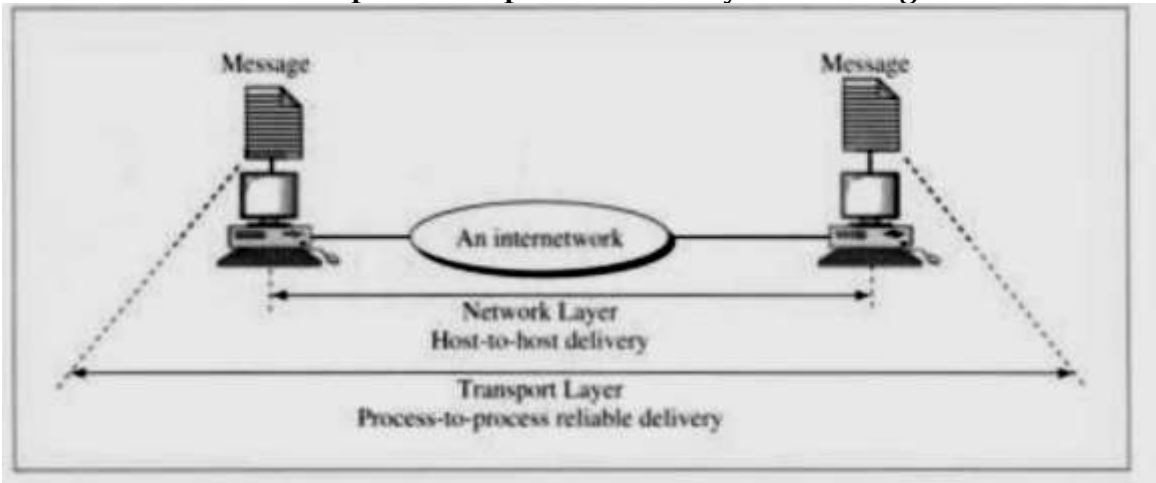
However, flow control at this layer is performed end lo end rather than across a single link.

■ **Error control:**

Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed end to end rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error(damage, loss or duplication). Error correction is usually achieved through retransmission.

Following Figure illustrates a process-to-process delivery by the transport layer

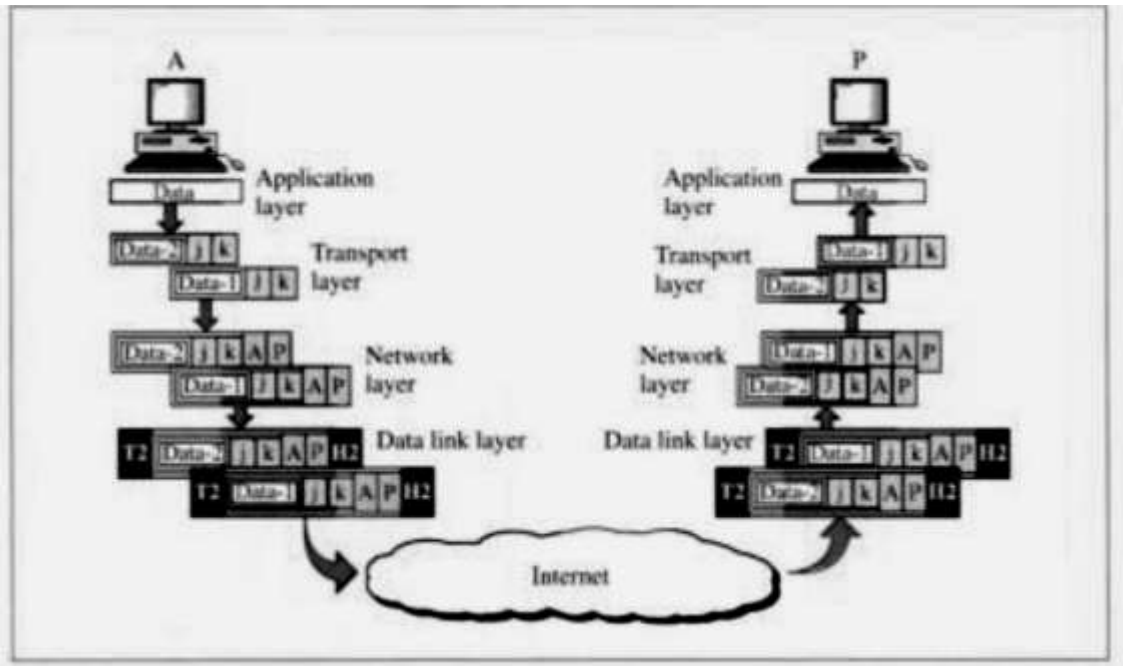
Reliable process to process delivery of a message



Example

Following Figure shows an example of transport layer communication .data coming from the upper layers have port addresses j and k (j is the address of the sending process. and k is the address of the receiving process). Since the data size is larger than the network layer can handle, the data are split into two packet, each packet retaining the port addresses (j and k). Then in the network layer, network addresses (A and P) are added in each packet .The packet can travel on different paths .and arrive at the destination either in order or out of order. The two packets are delivered in the destination transport layer, which is responsible for removing the transport layer headers and combining the two pieces of data for delivery to the application layer.

Example



➢ Session layer:

The Service provided by physical, data link and network layer are not sufficient for some processes. The session layer is network dialog controller
It was designed to establish, maintain, and synchronize the interaction between communicating systems.

Specific responsibilities of the session layer are as follows

1. **Dialog Control:**

The Session layer allows two systems to enter into a dialog. It allows communication between two processes to takes place either in half duplex or full duplex for Example:- dialog between a terminal connected to mainframe can be half duplex.

2. **Synchronization:**

The session layer allows a process to add checkpoints into a stream of data for Example if a system is sending a file of 200 pages, it is advisable to insert checkpoints .after every 100 pages to ensure that each 100 page unit is received and acknowledge independent .In these case if a crash happens during the transmission of pages 525 , retransmission begins at page 501,pages 1 tp 500 need not be retransmit

➢ Presentation Layer:

The presentation layer was designed to handle the syntax and semantics of the information exchanged between the two systems- It was designed for data translation, encryption, decryption, and compression.

1. Translation:

   The Processes (running program) in two systems are usually exchanging information in the form of character string ,numbers and so on. The information must be changed to bit stream before being transmitted. Because computers use different encoding system, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the information from its sender dependent format into a common format the presentation layer at the receiving machine changes the common format. The presentation layer at the receiving machine changes the common format into its receiver – dependent format.

2. **Encryption:**

   To carry sensitive information , the system must insure the privacy . Encryption means that the sender must transform the original information to another form and sends the resulting message out over the network . Decryption means reverse the original process to transform the message back to it's original form .

3. **Compression:**

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

➢ Application layer:

The application layer enables the user, whether human or software, to access the network It provides user interfaces and support for services such as electronic mail, remote file access and transfer, access to the World Wide Web. and so on.
Following Figure shows the relationship of the application layer to the user and the transport layer.

The major duties of the application layer are as follows:

■ **Mail services:**

This application is the basis for email forwarding and storage.

■ **file transfer and access.**

This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.

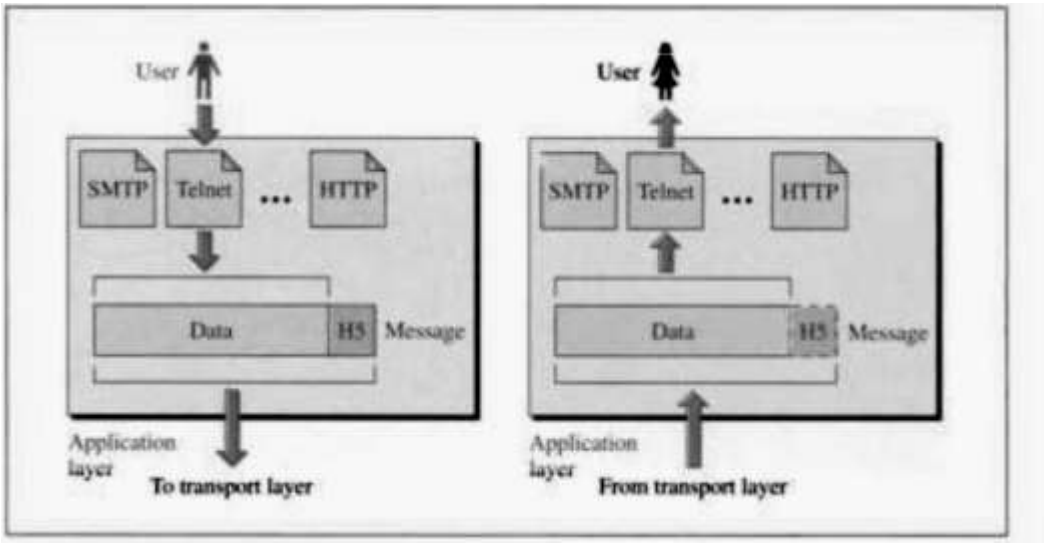
■ **Remote log-in:**

A user can log into a remote computer and access the resources of that computer.

■ **Accessing the World Wide Web:**

The most common application today is the access of the World Wide Web (WWW).

Application Layer



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

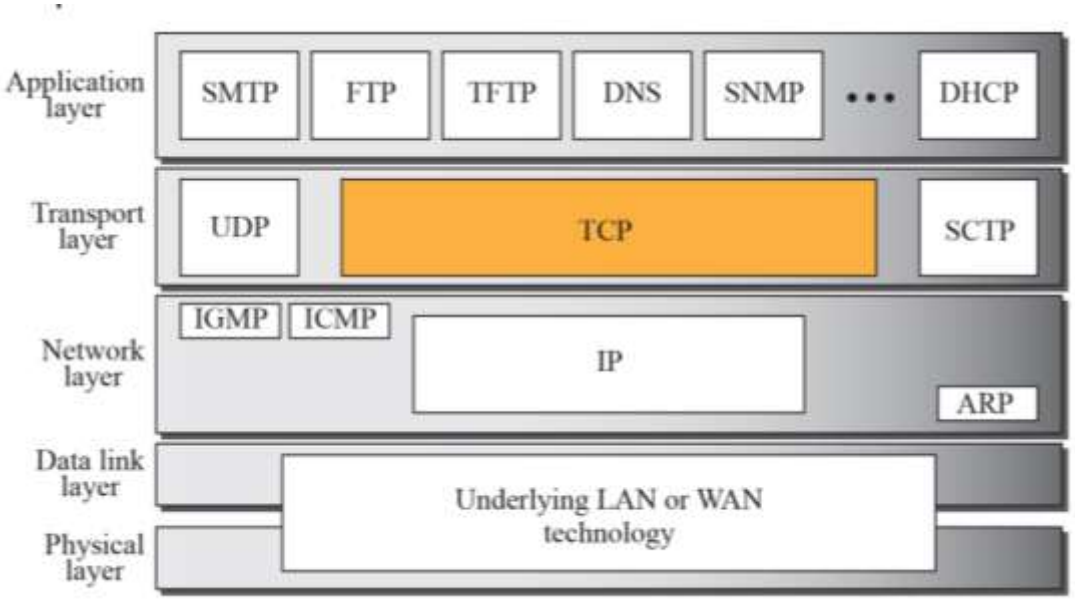
Summary of Layers

Following Figure summarizes the duties of each layer

Application	To allow access to network resources	7
Presentation	To translate, encrypt, and compress data	6
Session	To establish, manage, and terminate sessions	5
Transport	To provide reliable process-to-process message delivery and error recovery	4
Network	To move packets from source to destination; to provide internetworking	3
Data link	To organize bits into frames; to provide hop-to-hop delivery	2
Physical	To transmit bits over a medium; to provide mechanical and electrical specifications	1

1.3 TCP/IP Protocol Suite

TCP/IP PROTOCOL SUITE



TCP/IP PROTOCOL SUITE

Communications between computers on a network is done through protocol suits. The most widely used and most widely available protocol suite is TCP/IP protocol suite. A protocol suit consists of a layered architecture where each layer depicts some functionality which can be carried out by a protocol. Each layer usually has more than one protocol options to carry out the responsibility that the layer adheres to. TCP/IP is normally considered to be a 4 layer system. The 4 layers are as follows :

1. Application layer
2. Transport layer
3. Network layer
4. Data link layer

1. Application layer

This is the top layer of TCP/IP protocol suite. This layer includes applications or processes that use transport layer protocols to deliver the data to destination computers.
At each layer there are certain protocol options to carry out the task designated to that particular layer. So, application layer also has various protocols that applications use to communicate with the second layer, the transport layer. Some of the popular application layer protocols are :

- HTTP (Hypertext transfer protocol)
- FTP (File transfer protocol)
- SMTP (Simple mail transfer protocol)
- SNMP (Simple network management protocol) etc

2. Transport Layer

This layer provides backbone to data flow between two hosts. This layer receives data from the application layer above it. There are many protocols that work at this layer but the two most commonly used protocols at transport layer are TCP and UDP.
TCP is used where a reliable connection is required while UDP is used in case of unreliable connections.
TCP divides the data(coming from the application layer) into proper sized chunks and then passes these chunks onto the network. It acknowledges received packets, waits for the acknowledgments of the packets it sent and sets timeout to resend the packets if acknowledgements are not received in time. The term 'reliable connection' is used where it is not desired to loose any information that is being transferred over the network through this connection. So, the protocol used for this type of connection must provide the mechanism to achieve this desired characteristic. For example, while downloading a file, it is not desired to loose any information(bytes) as it may lead to corruption of downloaded content.
UDP provides a comparatively simpler but unreliable service by sending packets from one host to another. UDP does not take any extra measures to ensure that the data sent is received by the target host or not. The term 'unreliable connection' are used where loss of some information does not hamper the task being fulfilled through this connection. For example while streaming a video, loss of few bytes of information due to some reason is acceptable as this does not harm the user experience much.

3. Network Layer

This layer is also known as Internet layer. The main purpose of this layer is to organize or handle the movement of data on network. By movement of data, we generally mean routing of data over the network. The main protocol used at this layer is IP. While ICMP(used by popular 'ping' command) and IGMP are also used at this layer.

4. Data Link Layer

This layer is also known as network interface layer. This layer normally consists of device drivers in the OS and the network interface card attached to the system. Both the device drivers and the network interface card take care of the communication details with the media being used to transfer the data over the network. In most of the cases, this media is in the form of cables. Some of the famous protocols that are used at this layer include ARP(Address resolution protocol), PPP(Point to point protocol) etc.

5.Physical Network Layer

The **physical network layer** specifies the characteristics of the hardware to be used for the network. For example, physical network layer specifies the physical characteristics of the communications media. The physical layer of TCP/IP describes hardware standards such as IEEE 802.3, the specification for Ethernet network media, and RS-232, the specification for standard pin connectors.

SYBSCIT

Internet Technology

Chapter 2

Network Layer

2.1 Introduction to the Network Layer

- The main task of Network layer is to gets data from a source network to the destination network.
- This generally involves routing the packets across a network of networks (also known as internetwork).
- This is where IP(Internet Protocol) comes in. IP performs the basic task of getting packets of data from source to destination.
- The packets may require to make many hops at the intermediate routers while reaching the destination.
- This is the lowest layer that deals with end to end transmission.
- In order to achieve its goals, the network layer must know about the topology of 3the communication network.
- It must also take care to choose routes to avoid overloading of some of the communication lines while leaving others idle
- The functions of this layer include :
    1. Routing - The process of transferring packets received from the Data Link Layer of the source network to the Data Link Layer of the correct destination network is called routing. Involves decision making at each intermediate node on where to send the packet next so that it eventually reaches its destination. The node which makes this choice is called a router.

2. Inter-networking - The network layer is the same across all physical networks .Thus, if two physically different networks have to communicate, the packets that arrive at the Data Link Layer of the node which connects these two physically different networks, would be stripped of their headers and passed to the Network Layer. The network layer would then pass this data to the Data Link Layer of the other physical network.
3. Congestion Control - If the incoming rate of the packets arriving at any router is more than the outgoing rate, then congestion is said to occur.

2.2 Network layer services

1. It translates logical network address into physical address. Concerned with circuit, message or packet switching.
2. Routers and gateways operate in the network layer. Mechanism is provided by Network Layer for routing the3 packets to final destination.
3. Connection services are provided including network layer flow control, network layer error control and packet sequence control.
4. Breaks larger packets into small packets.
5. Transport packet from sending to receiving hosts via internet ‰
6. Network layer protocols exist in every host and router ‰
7. Three important functions:
8. Path determination: route taken by packets from source to destination (Routing algorithms)
9. Forwarding: move packets from a router's input to a appropriate router's output
10. Call setup: some networks require router call setup along path before data flows (e.g., MPLS)
11. Some of the Basic services of this layer include :

    1. Routing - The process of transferring packets received from the Data Link Layer of the source network to the Data Link Layer of the correct destination network is called routing. Involves decision making at each intermediate node on where to send the packet next so that it eventually reaches its destination. The node which makes this choice is called a router.
    2. Inter-networking - The network layer is the same across all physical networks .Thus, if two physically different networks have to communicate, the packets that arrive at the Data Link Layer of the node which connects these two physically different networks, would be stripped of their headers and passed to the Network Layer. The network layer would then pass this data to the Data Link Layer of the other physical network..
    3. Congestion Control - If the incoming rate of the packets arriving at any router is more than the outgoing rate, then congestion is said to occur.

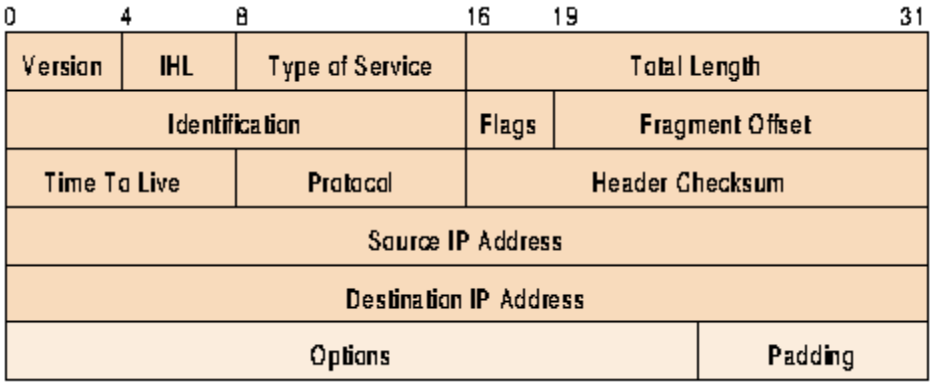
2.2.1 IPV4

- By default, the router you use will assign each of your computers their own IP address, often using NAT to forward the data coming from those computers to outside networks such as the Internet.
- If you need to register an IP address that can be seen on the Internet, you must register through InterNIC or use a web host that can assign you addresses.
- Every machine on a network has a unique identifier. Just as you would address a letter to send in the mail, computers use the unique identifier to send data to specific computers on a network.
- Most networks today, including all computers on the Internet, use the TCP/IP protocol as the standard for how to communicate on the network.
- In the TCP/IP protocol, the unique identifier for a computer is called its IP address
- The Internet Protocol Address (or IP Address) is a unique address that computing devices such as personal computers, tablets, and smartphones use to identify itself and communicate with other devices in the IP network.
- Any device connected to the IP network must have a unique IP address within the network. An IP address is analogous to a street address or telephone number in that it is used to uniquely identify an entity.
- The IP address identifies a system's location on the network in the same way a street address identifies a house on a city block.
- Just as a street address must identify a unique residence, an IP address must be globally unique to the internetwork and have a uniform format.

➢ **Internet Protocol (IP)**

- Internet Protocol is connectionless and unreliable protocol. It ensures no guarantee of successfully transmission of data.
- In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.
- IP provides a standard set of rules for sending and receiving data over the Internet.
- It allows devices running on different platforms to communicate with each other as long as they are connected to the Internet.

➢ **IP Packet Structure**



Version: 4 bits

- The Version field indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

- Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

- The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network.
- Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic.
- The major choice is a three way tradeoff between low-delay, high-reliability, and high-throughput.

Total Length: 16 bits

- Total Length is the length of the datagram, measured in octets, including internet header and data.
- This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks.
- All hosts must be prepared to accept datagrams of up to 576 octets
- It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.
- The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information

Identification: 16 bits

- An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

- Various Control Flags

Fragment Offset: 13 bits

- This field indicates where in the datagram this fragment belongs.
- The fragment offset is measured in units of 8 octets (64 bits).
- The first fragment has offset zero.

Time to Live: 8 bits

- This field indicates the maximum time the datagram is allowed to remain in the internet system.
- If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing.
- The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist.
- The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Protocol: 8 bits

- This field indicates the next level protocol used in the data portion of the internet datagram.
- The values for various protocols are specified in "Assigned Numbers" [9].

Header Checksum: 16 bits

- A checksum on the header only.
- Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

- The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header.
- For purposes of computing the checksum, the value of the checksum field is zero.
- This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

- The source address.

Destination Address: 32 bits

- The destination address.

Options: variable

- The options may appear or not in datagrams.
- They must be implemented by all IP modules (host and gateways).
- What is optional is their transmission in any particular datagram, not their implementation.
- In some environments the security option may be required in all datagrams.
- The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:

Case 1: A single octet of option-type.
Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Each IP address includes a network ID and a host ID.

Network ID

- Network ID is also known as a network address that identifies the systems that are located on the same physical network bounded by IP routers.
- All systems on the same physical network must have the same network ID.
- The network ID must be unique to the internetwork.

Host ID

- Host ID is also known as a host address that identifies a workstation, server, router, or other TCP/IP host within a network.
- The host address must be unique to the network ID

IPv4 Address Syntax

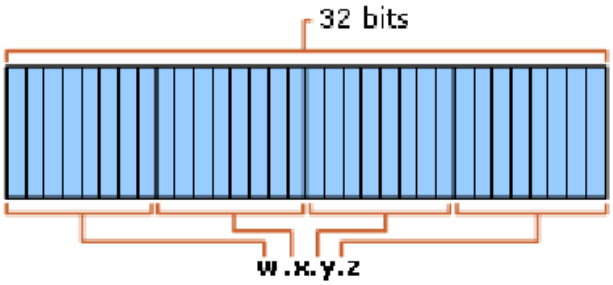
- An IP address consists of 32 bits.
- Instead of expressing IPv4 addresses 32 bits at a time using binary notation , it is standard practice to segment the 32 bits of an IPv4 address into four 8-bit fields called *octets*.
- Each octet is converted to a decimal number (base 10) from 0–255 and separated by a period (a dot).
- This format is called *dotted decimal notation*.

An IP Ad3dress in Binary and Dotted Decimal Formats

Binary Format	Dotted Decimal Notation
11000000 10101000 00000011 00011000	192.168.3.24

1. Segmented into 8-bit blocks: 11000000 10101000 00000011 00011000.
2. Each block is converted to decimal: 192 168 3 243
3. The adjacent octets are separated by a period: 192.168.3.24.

IP Address

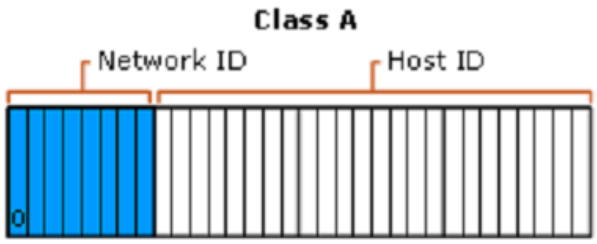


2.2.2 Internet Address Classes

- The Internet community originally defined address classes to accommodate different types of addresses and networks of varying sizes.
- The class of address defined which bits were used for the network ID and which bits were used for the host ID.
- It also defined the possible number of networks and the number of hosts per network. Of five address classes, class A, B, and C addresses were defined for IPv4 unicast addresses.
- Class D addresses were defined for IPv4 multicast addresses and class E addresses were defined for experimental uses.

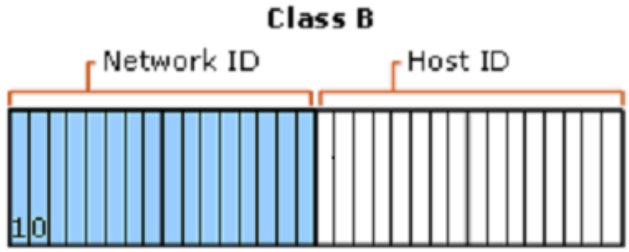
Class A

- Class A network IDs were assigned to networks with a very large number of hosts.
- The high-order bit in a class A address is always set to zero, which makes the address prefix for all class A networks and addresses 0.0.0.0/1 or 128.0.0.0.
- The next seven bits (completing the first octet) are used to enumerate class A network IDs.
- Therefore, address prefixes for class A network IDs have an 8-bit prefix length (/8 or 255.0.0.0).
- The remaining 24 bits (the last three octets) are used for the host ID. The address prefix 0.0.0.0/0 or 0.0.0.0, 0.0.0.03 is a reserved network ID and 127.0.0.0/8 or 127.0.0.0 or 255.0.0.03 is reserved for loopback addresses.
- Out of a total of 128 possible class A networks, there are 126 networks and 16,777,214 hosts per network.



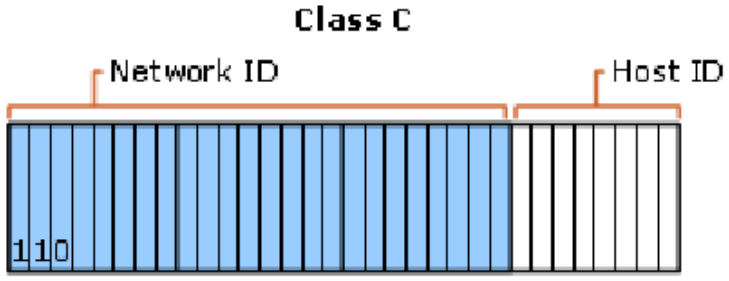
Class B

- Class B network IDs were assigned to medium to large-sized networks.
- The two high-order bits in a class B address are always set to 10, which makes the address prefix for all class B networks and addresses 128.0.0.0/2 or 128.0.0.0 or 192.0.0.0.
- The next 14 bits (completing the first two octets) are used to enumerate class B network IDs.
- Therefore, address prefixes for class B network IDs have a 16-bit prefix length (/16 or 255.255.0.0).
- The remaining 16 bits (last two octets) are used for the host ID. With 14 bits to express class B network IDs and 16 bits to express host IDs, this allows for 16,384 networks and 65,534 hosts per network.



Class C

- This addresses were assigned to small networks.
- The three high-order bits in a class C address are always set to 110, which makes the address prefix for all class C networks and addresses 192.0.0.0/3 or 192.0.0.0 or 224.0.0.0.
- The next 21 bits (completing the first three octets) are used to enumerate class C network IDs.
- Therefore, address prefixes for class C network IDs have a 24-bit prefix length (/24 or 255.255.255.0).
- The remaining 8 bits (the last octet) are used for the host ID. With 21 bits to express class C network IDs and 8 bits to express host IDs, this allows for 2,097,152 networks and 254 hosts per network.



Class D

- Class D addresses are reserved for IPv4 multicast addresses.
- The four high-order bits in a class D address are always set to 1110, which makes the address prefix for all class D addresses 224.0.0.0/4 or 224.0.0.0

Class E

- Class E addresses are reserved for experimental use.
- The high-order bits in a class E address are set to 1111, which makes the address prefix for all class E addresses 240.0.0.0/4 or 240.0.0.0

Special IPv4 Addresses

1. **0.0.0.0**

- Known as the unspecified IPv4 address, it is used to indicate the absence of an address.
- The unspecified address is used only as a source address when the IPv4 node is not configured with an IPv4 address configuration and is attempting to obtain an address through a configuration protocol such as Dynamic Host Configuration Protocol (DHCP).

2. **127.0.0.1**

- Known as the IPv4 loopback address, it is assigned to an internal loopback interface, enabling a node to send packets to itself.

2.2.3 IPV6 Addresses and Protocol

- An IPv6 address is made of 128 bits divided into eight 16-bits blocks. Each block is then converted into 4-digit Hexadecimal numbers separated by colon symbols.
- IPv6 addresses are denoted by eight groups of hexadecimal quartets separated by colons in between them.
- Following is an example of a valid IPv6 address: 2001:cdba:0000:0000:0000:0000:3257:9652
- Any four-digit group of zeroes within an IPv6 address may be reduced to a single zero or altogether omitted.
- Therefore, the following IPv6 addresses are similar and equally valid:
  2001:cdba:0000:0000:0000:0000:3257:9652
  2001:cdba:0:0:0:0:3257:9652
  2001:cdba::3257:9652

➢ **Network Notation in IPv6**

- The IPv6 networks are denoted by Classless Inter Domain Routing (CIDR) notation.
- A network or subnet using the IPv6 protocol is denoted as a contiguous group of IPv6 addresses whose size must be a power of two.
- The initial bits of an IPv6 address (these are identical for all hosts in a network) form the networks prefix. The size of bits in a network prefix are separated with a /.

- For example, 2001:cdba:9abc:5678::/64 denotes the network address 2001:cdba:9abc:5678. This network comprises of addresses rearranging from 2001:cdba:9abc:5678:: up to 2001:cdba:9abc:5678:ffff:ffff:ffff:ffff. In a similar fashion, a single host may be denoted as a network with a 128-bit prefix.
- In this way, IPv6 allows a network to comprise of a single host and above.
  -

➢ **IPv6 addresses are broadly classified into three categories:**

1**) Unicast addresses**

- A Unicast address acts as an identifier for a single interface.
- An IPv6 packet sent to a Unicast address is delivered to the interface identified by that address.

2) Multicast addresses

- A Multicast address acts as an identifier for a group/set of interfaces that may belong to the different nodes.
- An IPv6 packet delivered to a Multicast address is delivered to the multiple interfaces.

3) Anycast addresses

- Anycast addresses act as identifiers for a set of interfaces that may belong to the different nodes.
- An IPv6 packet destined for an Anycast address is delivered to one of the interfaces identified by the address.

➢ **IPv6 protocol**

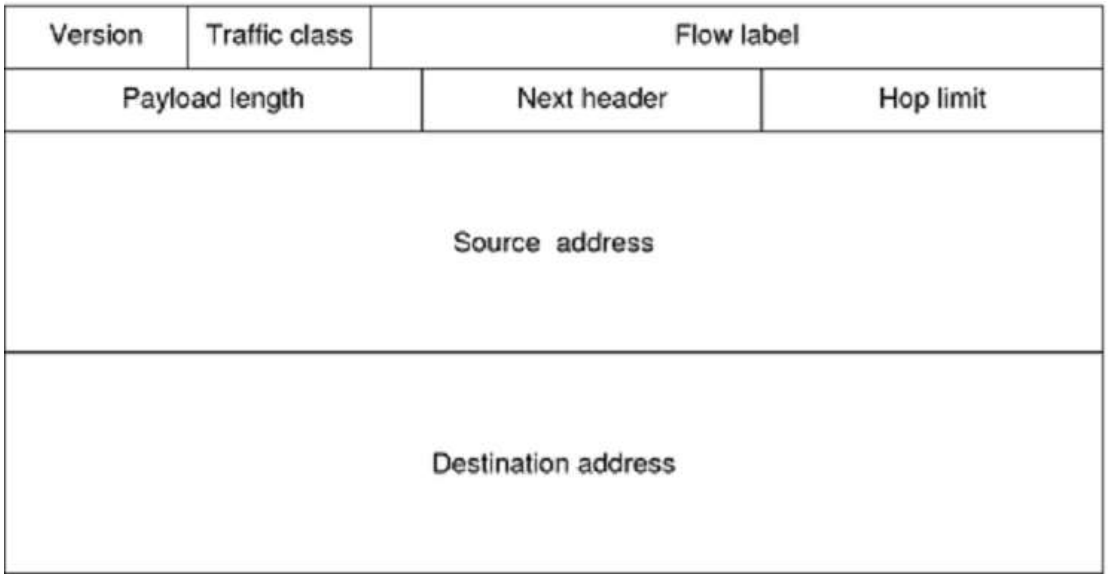
- The IPv6 protocol component that is installed in Windows operating systems is a series of interconnected protocols that include Internet Control Message Protocol version 6 (ICMPv6), Multicast Listener Discovery (MLD), and Neighbor Discovery.
- These core protocols replace the Internet layer protocols in the Defense Advanced Research Projects Agency (DARPA) model.
- All protocols above the Internet layer rely on the basic services that IPv6 provides.
- Protocols at the Host-to-Host Transport and Application layers are largely unchanged, except when addresses are part of the payload or part of the data structures that the protocol maintains.
- For example, both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) must be updated to perform new checksum calculations that include IPv6 addresses.
- TCP must be updated to store IPv6 addresses in its internal Transmission Control Block (TCB). Routing Information Protocol (RIP) must be updated to send and receive IPv6 route prefixes.

➢ IPv6 Packet

IP Header	Extension Header	Upper Layer Protocol

- We may divide IPv6 datagram packet header as three parts.
- 1) IPv6 datagram packet header
- 2) Extension Header
- 3) Upper Layer Protocol Data.
- IPv6 datagram packet has also extension headers of varying lengths.
- If extension headers are present in IPv6 datagram packet, a Next Header field in the IPv6 header points the first extension header.
- Each extension header contains another Next Header field, pointing the next extension header.
- The last IPv6 datagram packet extension header points the upper layer protocol header (Transmission Control Protocol (TCP), User Datagram Protocol (UDP) , or Internet Control Message Protocol (ICMPv6)).

➢ IPv6 Header Format



• **Version:** The size of the Version field is 4 bits. The Version field shows the version of IP and is set to 6.

• **Traffic Class:** The size of Traffic Class field is 8 bits. Traffic Class field is similar to the IPv4 Type of Service (ToS) field. The Traffic Class field indicates the IPv6 packet's class or priority.

• **Flow Label:** The size of Flow Label field is 20 bits. The Flow Label field provide additional support for real-time datagram delivery and quality of service features. The purpose of Flow Label field is to indicate that this packet belongs to a specific sequence of packets between a source and destination and can be used to prioritized delivery of packets for services like voice.

• **Payload Length:** The size of the Payload Length field is 16 bits. The Payload Length field shows the length of the IPv6 payload, including the extension headers and the upper layer protocol data

• **Next Header:** The size of the Next Header field is 8 bits. The Next Header field shows either the type of the first extension (if any extension header is available) or the protocol in the upper layer such as TCP, UDP, or ICMPv6.

• **Hop Limit:** The size of the Hop Limit field is 8 bits The Hop Limit field shows the maximum number of routers the IPv6 packet can travel. This Hop Limit field is similar to IPv4 Time to Live (TTL) field.

This field is typically used by distance vector routing protocols, like Routing Information Protocol (RIP) to prevent layer 3 loops (routing loops).

• **Source Address:** The size of the Source Address field is 128 bits. The Source Address field shows the IPv6 address of the source of the packet.

• **Destination Address:** The size of the Destination Address field is 128 bits. The Destination Address field shows the IPv6 address of the destination of the packet.

2.3 packet switching

- Packet switching can be used as an alternate to circuit switching.
- In the packet switched networks, data is sent in discrete units that have variable length. They are called as packets.
- There is a strict upper bound limit on the size of packets in a packet switch network.
- The packet contains data and various control information.
- The packet switched networks allow any host to send data to any other host without reserving the circuit.
- Multiple paths between a pair of sender and receiver may exist in a packet switched network.
- One path is selected between source and destination. Whenever the sender has data to send, it converts them into packets and forwards them to next computer or router.
- The router stores this packet till the output line is free.
- Then, this packet is transferred to next computer or router (called as hop). This way, it moves to the destination hop by hop.
- All the packets belonging to a transmission may or may not take the same route. The route of a packet is decided by network layer protocols.
- In packet switching, data will be transmitted by carry forward method that means it will go to the nearest station from there it will go to the destination.
- 3Network layer is responsible for routing in packet switching
- Packet-switched networks are classified as connectionless networks and connection-oriented networks, depending on the technique used for transferring information.
- The simplest form of a network service is based on the connectionless protocol that does not require a call setup prior to transmission of packets.

- A related, though more complex, service is the connection-oriented protocol in which packets are transferred through an established virtual circuit between a source and a destination.

A. Connectionless Networks

- Connectionless networks is also refer as datagram networks.
- In this networking approach, a large piece of data is normally fragmented into smaller pieces, and then each piece of data is encapsulated into a certain 3“formatted” header, resulting in the basic Internet transmission packet, or datagram.
- We interchangeably use packets and datagrams for connectionless networks.
- Packets from a source are routed independently of one another.
- In this type of network, a user can transmit a packet anytime, without notifying the network layer.
- A packet is then sent over the network, with each router receiving the packet forwarding it to the best router it knows, until the packet reaches the destination.
- The connectionless networking approach does not require a call setup to transfer packets, but it has error-detection capability.
- The main advantage of this scheme is its capability to route packets through an alternative path in case a fault is present on the desired transmission link.
- On the flip side, since packets belonging to the same source may be routed independently over different paths, the packets may arrive out of sequence; in such a case, the misordered packets are resequenced and delivered to the destination.

B. Connection-Oriented Networks

- In 3connection-oriented networks, or virtual-circuit networks, a route setup between a source and a destination is required prior to data transfer, as in the case of conventional telephone networks.
- In this networking scheme, once a connection or a path is initially set up, network resources are reserved for the communication duration, and all packets belonging to the same source are routed over the established connection.
- After the communication between a source and a destination is finished, the connection is terminated using a connection-termination procedure.
- During the call setup, the network can offer a selection of options, such as best-effort service, reliable service, guaranteed delay service, and guaranteed bandwidth service

Advantages of packet Switching

- The main advantage of packet switching is the efficiency of the network. In circuit switching network, a reserved circuit cannot be used by others, till the sender and receiver leave it. Even if no data is being sent on a reserved circuit, no one else can access the circuit. This results in network bandwidth wastage. The packet switching reduces network bandwidth wastage.
- The other advantage is that the packet switching is more faults tolerant. In case of circuit switching, all the packets are lost if a router in the circuit is down as all the packets follow the same route. But, in case of packet switching network, the packets can be routed over the malfunctioning component of the network. This is because all the packets may follow a different route to the destination.
- The circuit switching bills the user depending on the distance and duration of connection whereas packet switching network bill users only on the basis of duration of connectivity.

- The advantage of circuit switching network over packet switching network is that the circuit switching network provides ordered delivery of packets. As all the packets follow the same route. They arrive in correct order at destination.
- It uses a digital network. This method enables digital data to be directly transmitted to a destination, and is therefore appropriate for data communication systems.
- High data transmission quality - The quality of data transmission in a packet switched network is kept high (error free) because the data distribution is checked and error detection is employed during data transmission.

Disadvantages of Packet Switching

- Packets m3ay be lost on their route, so sequence numbers are required to identify missing packets.
- Switching nodes requires more processing power as the packet switching protocols are more complex.
- Switching nodes for packet switching require large amount of RAM to handle large quantities of packets.
- A significant data transmission delay occurs - Use of store and forward method causes a significant data transmission.

2.4 network layer performance

- For a given network, one might be interested to know how well it is performing.
- One might also wish to know what could be done to further improve the performance, or if the network is giving the peak performance.
- Thus, one needs to do a comparative study of the network by considering different options.
- This performance evaluation helps the user to determine the suitable network configuration that serves him best.
- For example consider a new startup organization which has setup its own web portal.
- As the portal gradually becomes popular then network traffic increases which would degrade its performance.
- Therefore, one should have a well configured network with proper load balancing capabilities.
- Before we can proceed with performance evaluation, we must choose the different metrics that would help us in making comparisons.
- There could be different metrics to determine the performance like throughput, delay, jitter, packet loss.
- The choice of metric would depend upon the purpose the network has been setup for.
- The metrics could be related to the different layers of the network stack.
- Let's take a look at the most common performance measurement terms and see what they are all about.

Speed

- This is the most generic performance term used in networking. As such, it can mean just about anything. Most commonly, however, it refers to the rated or nomina lspeed of a particular networking technology.
  For example, Fast Ethernet has a nominal speed of 100 megabits per second; it is for that reason often called 100 Mbit Ethernet, or given a designation such as “100BASE-TX”.
  Rated speed is the biggest “performance magic number” in networking—you see it used to label hardware devices, and many people bandy the numbers about as if they actually were the real “speed of the network”.

Bandwidth

- Bandwidth is a widely-used term that usually refers to the data-carrying capacity of a network or data transmission medium.
- It indicates the maximum amount of data that can pass from one point to another in a unit of time.
- The term comes from the study of electromagnetic radiation, where it refers to the width of a band of frequencies used to carry data. It is usually given in a theoretical context, though not always.
- Bandwidth is still used in these two senses: "frequency band width" and data capacity.

Latency

- It can take a long time for a packet to be delivered across intervening networks.
- In reliable protocols where a receiver acknowledges delivery of each chunk of data, it is possible to measure this as round-trip time.

Packet loss

- In some cases, intermediate devices in a network will lose packets.
- This may be due to errors, to overloading of the intermediate network, or to intentional discarding of traffic in order to enforce a particular service level.

Retransmission

- When packets are lost in a reliable network, they are retransmitted.
- This incurs two delays: First, the delay from re-sending the data; and second, the delay resulting from waiting until the data is received in the correct order before forwarding it up the protocol stack.

Throughput

- The amount of traffic a network can carry is measured as throughput, usually in terms such as kilobits per second.
- Throughput is analogous to the number of lanes on a highway, whereas latency is analogous to its speed limit.

Jitter

- Jitter is the undesired deviation from true periodicity of an assumed periodic signal in electronics and telecommunications, often in relation to a reference clock source.
- Jitter may be observed in characteristics such as the frequency of successive pulses, the signal amplitude, or phase of periodic signals.
- 3Jitter is a significant, and usually undesired, factor in the design of almost all communications links (e.g., USB, PCI-e, SATA, OC-48). In clock recovery applications it is called timing jitter.[

Error rate

- In digital transmission, the number of bit errors is the number of received bits of a data stream over a communication channel that have been altered due to noise, interference, distortion or bit synchronization errors.
- The bit error rate or bit error ratio (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. BER is a unitless performance measure, often expressed as a percentage

2.5 IPv4 addressing

- By default, the router you use will assign each of your computers their own IP address, often using NAT to forward the data coming from those computers to outside networks such as the Internet.
- If you need to register an IP address that can be seen on the Internet, you must register through InterNIC or use a web host that can assign you addresses.
- Every machine on a network has a unique identifier. Just as you would address a letter to send in the mail, computers use the unique identifier to send data to specific computers on a network.
- Most networks today, including all computers on the Internet, use the TCP/IP protocol as the standard for how to communicate on the network.
- In the TCP/IP protocol, the unique identifier for a computer is called its IP address
- The Internet Protocol Address (or IP Address) is a unique address that computing devices such as personal computers, tablets, and smart phones use to identify itself and communicate with other devices in the IP network.
- Any device connected to the IP network must have a unique IP address within the network. An IP address is analogous to a street address or telephone number in that it is used to uniquely identify an entity.
- The IP address identifies a system's location on the network in the same way a street address identifies a house on a city block.
- Just as a street address must identify a unique residence, an IP address must be globally unique to the internetwork and have a uniform format.

Each IP address includes a network ID and a host ID.

Network ID

- Network ID is also known as a network address that identifies the systems that are located on the same physical network bounded by IP routers.
- All systems on the same physical network must have the same network ID.
- The network ID must be unique to the internetwork.

Host ID

- Host ID is also known as a host address that identifies a workstation, server, router, or other TCP/IP host within a network.
- The host address must be unique to the network ID

IPv4 Address Syntax

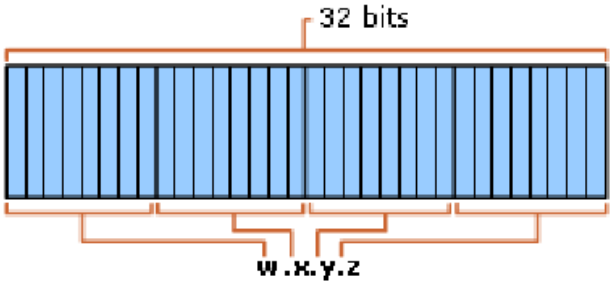
- An IP address consists of 32 bits.
- Instead of expressing IPv4 addresses 32 bits at a time using binary notation , it is standard practice to segment the 32 bits of an IPv4 address into four 8-bit fields called *octets*.
- Each octet is converted to a decimal number (base 10) from 0–255 and separated by a period (a dot).
- This format is called *dotted decimal notation*.

An IP Ad3dress in Binary and Dotted Decimal Formats

Binary Format	Dotted Decimal Notation
11000000 10101000 00000011 00011000	192.168.3.24

4. Segmented into 8-bit blocks: 11000000 10101000 00000011 00011000.
5. Each block is converted to decimal: 192 168 3 243
6. The adjacent octets are separated by a period: 192.168.3.24.

IP Address

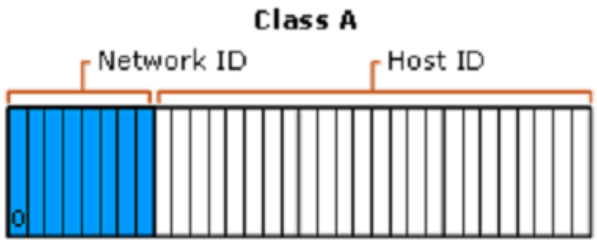


Internet Address Classes

- The Internet community originally defined address classes to accommodate different types of addresses and networks of varying sizes.
- The class of address defined which bits were used for the network ID and which bits were used for the host ID.
- It also defined the possible number of networks and the number of hosts per network. Of five address classes, class A, B, and C addresses were defined for IPv4 unicast addresses.
- Class D addresses were defined for IPv4 multicast addresses and class E addresses were defined for experimental uses.

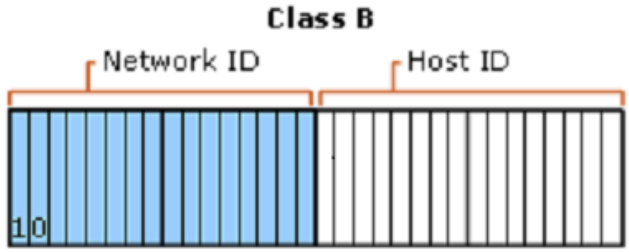
Class A

- Class A network IDs were assigned to networks with a very large number of hosts.
- The high-order bit in a class A address is always set to zero, which makes the address prefix for all class A networks and addresses 0.0.0.0/1 or 128.0.0.0.
- The next seven bits (completing the first octet) are used to enumerate class A network IDs.
- Therefore, address prefixes for class A network IDs have an 8-bit prefix length (/8 or 255.0.0.0).
- The remaining 24 bits (the last three octets) are used for the host ID. The address prefix 0.0.0.0/0 or 0.0.0.0, 0.0.0.03 is a reserved network ID and 127.0.0.0/8 or 127.0.0.0 or 255.0.0.03 is reserved for loopback addresses.
- Out of a total of 128 possible class A networks, there are 126 networks and 16,777,214 hosts per network.



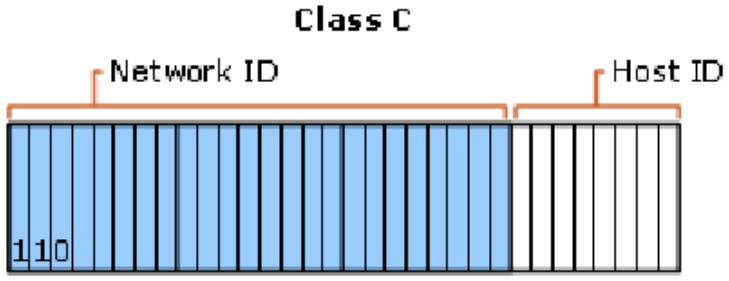
Class B

- Class B network IDs were assigned to medium to large-sized networks.
- The two high-order bits in a class B address are always set to 10, which makes the address prefix for all class B networks and addresses 128.0.0.0/2 or 128.0.0.0 or 192.0.0.0.
- The next 14 bits (completing the first two octets) are used to enumerate class B network IDs.
- Therefore, address prefixes for class B network IDs have a 16-bit prefix length (/16 or 255.255.0.0).
- The remaining 16 bits (last two octets) are used for the host ID. With 14 bits to express class B network IDs and 16 bits to express host IDs, this allows for 16,384 networks and 65,534 hosts per network.



Class C

- This addresses were assigned to small networks.
- The three high-order bits in a class C address are always set to 110, which makes the address prefix for all class C networks and addresses 192.0.0.0/3 or 192.0.0.0 or 224.0.0.0.
- The next 21 bits (completing the first three octets) are used to enumerate class C network IDs.
- Therefore, address prefixes for class C network IDs have a 24-bit prefix length (/24 or 255.255.255.0).
- The remaining 8 bits (the last octet) are used for the host ID. With 21 bits to express class C network IDs and 8 bits to express host IDs, this allows for 2,097,152 networks and 254 hosts per network.



Class D

- Class D addresses are reserved for IPv4 multicast addresses.
- The four high-order bits in a class D address are always set to 1110, which makes the address prefix for all class D addresses 224.0.0.0/4 or 224.0.0.0

Class E

- Class E addresses are reserved for experimental use.
- The high-order bits in a class E address are set to 1111, which makes the address prefix for all class E addresses 240.0.0.0/4 or 240.0.0.0

Special IPv4 Addresses

1. **0.0.0.0**

- Known as the unspecified IPv4 address, it is used to indicate the absence of an address.
- The unspecified address is used only as a source address when the IPv4 node is not configured with an IPv4 address configuration and is attempting to obtain an address through a configuration protocol such as Dynamic Host Configuration Protocol (DHCP).

2. **127.0.0.1**

- Known as the IPv4 loopback address, it is assigned to an internal loopback interface, enabling a node to send packets to itself.

2.6 Forwarding of IP packets

- Packet forwarding is the basic method for sharing information across systems on a network. Packets are transferred between a source interface and a destination interface, usually on two different systems.
- When you issue a command or send a message to a nonlocal interface, your system forwards those packets onto the local network.
- The interface with the destination IP address that is specified in the packet headers then retrieves the packets from the local network.
- If the destination address is not on the local network, the packets are then forwarded to the next adjacent network, or hop. By default, packet forwarding is automatically configured when you install Oracle Solaris.
- Routing is the process by which systems decide where to send a packet.
- Routing protocols on a system “discover” the other systems on the local network.
- When the source system and the destination system are on the same local network, the path that packets travel between them is called a direct route.
- If a packet must travel at least one hop beyond its source system, the path between the source system and destination system is called an indirect route.

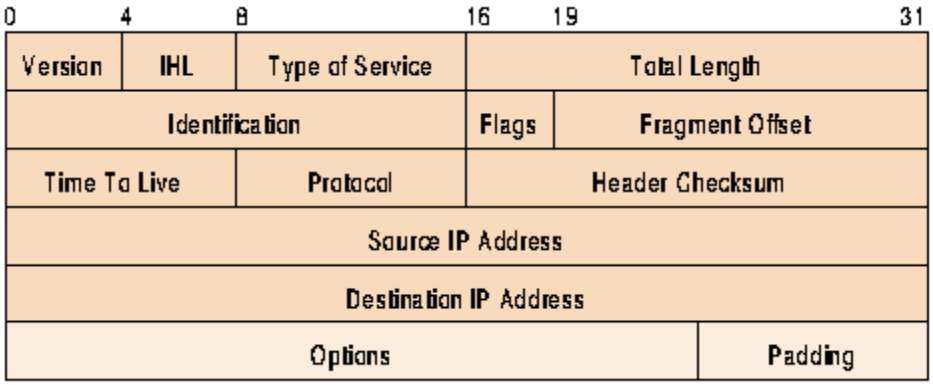
- The routing protocols learn the path to a destination interface and retain data about known routes in the system's routing table.
- Routers are specially configured systems with multiple physical interfaces that connect the router to more than one local network.
- Therefore, the router can forward packets beyond the home LAN, regardless of whether the router runs a routing protocol.
- For more information about how routers forward packets, refer to Planning for Routers on Your Network.
- Routing protocols handle routing activity on a system and, by exchanging routing information with other hosts, maintain known routes to remote networks.
- Both routers and hosts can run routing protocols. The routing protocols on the host communicate with routing daemons on other routers and hosts.
- These protocols assist the host in determining where to forward packets. When network interfaces are enabled, the system automatically communicates with the routing daemons.
- These daemons monitor routers on the network and advertise the routers' addresses to the hosts on the local network. Some routing protocols, though not all, also maintain statistics that you can use to measure routing performance.
- Unlike packet forwarding, you must explicitly configure routing on an Oracle Solaris system

2.7 Internet Protocol

Internet Protocol (IP)

- Internet Protocol is connectionless and unreliable protocol. It ensures no guarantee of successfully transmission of data.
- In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.
- IP provides a standard set of rules for sending and receiving data over the Internet.
- It allows devices running on different platforms to communicate with each other as long as they are connected to the Internet.

➢ IP Packet Structure



Version: 4 bits

- The Version field indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

- Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

- The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network.
- Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic.
- The major choice is a three way tradeoff between low-delay, high-reliability, and high-throughput.

Total Length: 16 bits

- Total Length is the length of the datagram, measured in octets, including internet header and data.
- This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks.
- All hosts must be prepared to accept datagrams of up to 576 octets
- It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.
- The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information

Identification: 16 bits

- An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

- Various Control Flags

Fragment Offset: 13 bits

- This field indicates where in the datagram this fragment belongs.

- The fragment offset is measured in units of 8 octets (64 bits).
- The first fragment has offset zero.

Time to Live: 8 bits

- This field indicates the maximum time the datagram is allowed to remain in the internet system.
- If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing.
- The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist.
- The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Protocol: 8 bits

- This field indicates the next level protocol used in the data portion of the internet datagram.
- The values for various protocols are specified in "Assigned Numbers" [9].

Header Checksum: 16 bits

- A checksum on the header only.
- Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

- The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header.
- For purposes of computing the checksum, the value of the checksum field is zero.
- This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

- The source address.

Destination Address: 32 bits

- The destination address.

Options: variable

- The options may appear or not in datagrams.
- They must be implemented by all IP modules (host and gateways).
- What is optional is their transmission in any particular datagram, not their implementation.
- In some environments the security option may be required in all datagrams.
- The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:
  Case 1: A single octet of option-type.
  Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

- In order for a Internet-connected host to be recognized by other devices, it must have an IP address.
- This may be either an IPv4 or IPv6 address, but either way it uniquely defines a device on the Internet.
- The Internet Protocol also provides basic instructions for transferring packets between devices.
- However, it does not actually establish the connection or define the ordering of the packets transmitted.
- These aspects are handled by the Transmission Control Protocol, which works in conjunction with the Internet Protocol to transfer data between systems on the Internet.
- For this reason, connections between Internet-connected systems are often called "TCP/IP" connections.

Internet protocols

- Several protocols are used on the Internet, including Electronic Mail (e-mail), File Transfer Protocol (FTP), HTTP (World Wide Web), News (or Usenet), Gopher and Telnet. Each of these has its own standard and usage.

Electronic Mail

- Included in the email protocol are three distinct protocols. SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol) and POP3 (Post Office Protocol 3).

- SMTP is a protocol used for sending mail, while IMAP and POP3 are used for receiving. Almost all Internet service providers support all three protocols.

- However the most popular setup for most providers is to use SMTP for sending mail while using POP3 for receiving.

File Transfer Protocol

- File Transfer Protocol, or FTP, is a means of transferring a file from one computer to another.

- FTP is commonly used for uploading a web page to a web server so that it may be seen on the World Wide Web.

- A special program, called a client, is usually needed to use FTP.

HTTP (World Wide Web)

- HyperText Transfer Protocol, or HTTP, is the protocol used by web server to allow web pages to be shown in a web browser.
- If you look up into the address bar of your web browser, the place where you type in the address that you want to visit, it has the prefix "http://" in front of the address.
- Because most web browsers are capable of FTP as well as viewing web pages, the http tells the browser what kind of information to expect.

News (or Usenet)

- Network News Transfer Protocol (NNTP) is used for serving Usenet posts Usenet is similar to the forums that many web sites have.
- Usenet has forums that are dedicated to specific companies as well as forums that have a wide range of topics. Usenet is divided into several areas.
- Some of the forums that are included in Usenet are comp. for discussion of computer-related topics, sci. for discussion of scientific subjects, rec.
- for discussion of recreational activities (e.g. games and hobbies) and talk. for discussion of contentious issues such as religion and politics.

Gopher

- Another tool of the Internet is Gopher, a menu-based program that enables you to browse for information without knowing where the material is located.
- It lets you search a list of resources and then sends the material to you

Telnet

- Telnet lets you log in to a remote computer just as you would if you were there.
- So any commands that you would be able to run from the remote computer if you were sitting in front of it, you would be able to run from the computer you logged in from.

SYBSCIT
Internet Technology
Chapter 3
ARP

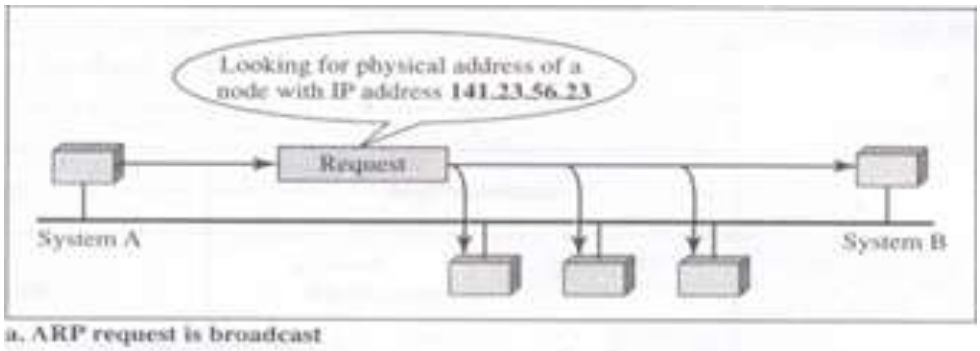
3.1 Address Resolution Protocol (ARP)

ARP

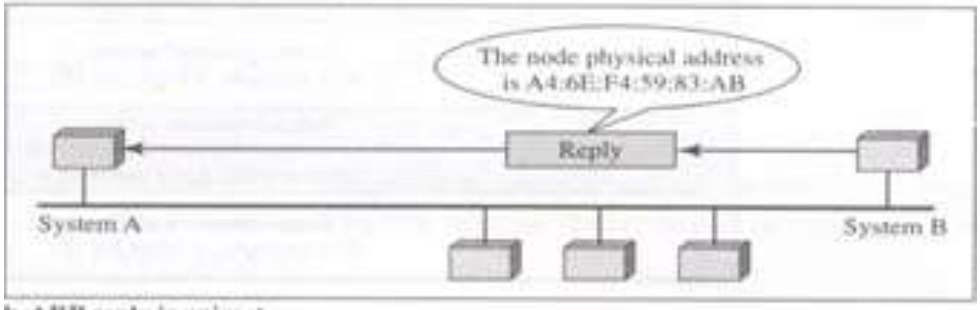
It is mapping Logical Address to Physical Address

Following 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.

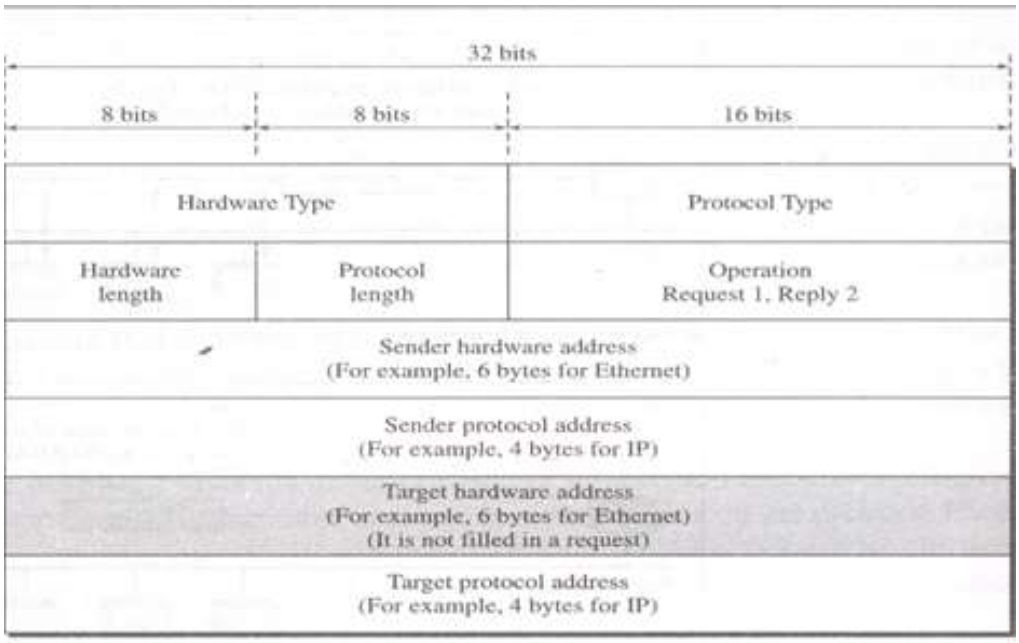


a. ARP request is broadcast



b. ARP reply is unicast

3.1.1ARP Packet Format



The fields are as follows:

HYPE (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.

PTYPE (Protocol type) This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 080016. ARP can be used with any higher-level protocol.

HLEN (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.

PLEN (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.

OPER (Operation) This is a 16-bit field defining the type of packet.
Two packet types are defined: ARP request (1) and ARP reply (2).

SHA (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.

SPA (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.

THA (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 all Os because the sender does not know the physical address of the target.

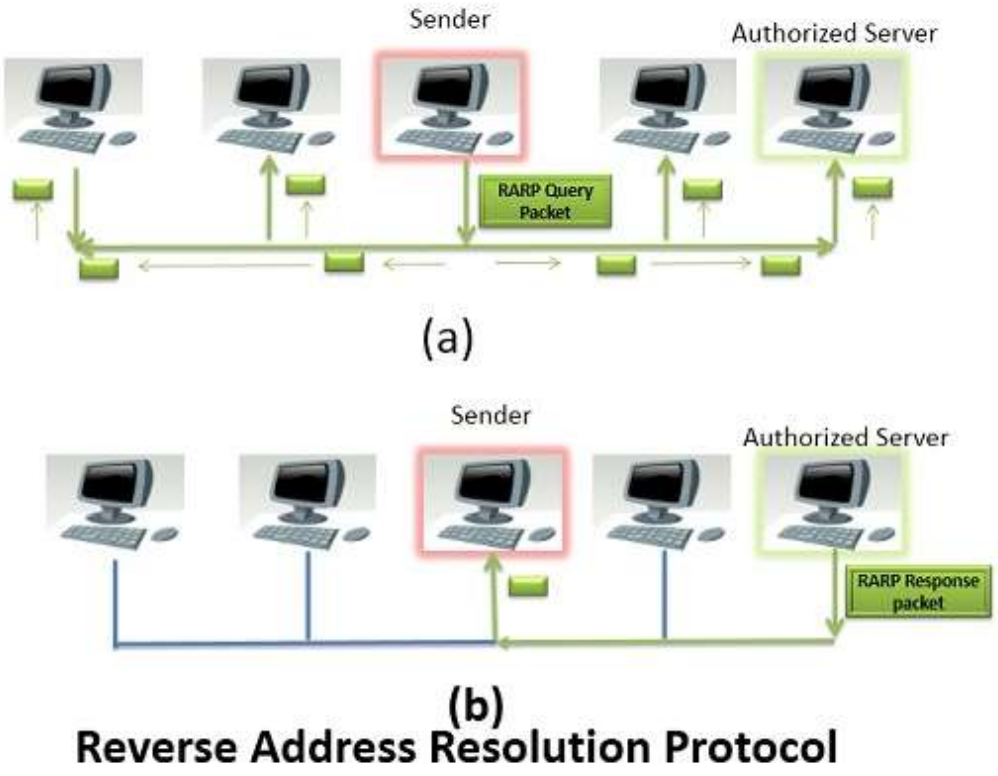
TPA (Target protocol address) This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

3.2 RARP

RARP (Reverse Address Resolution Protocol) is also a network layer protocol. RARP is a TCP/IP protocol that allows any host to obtain its IP address from the server. RARP is adapted from the ARP protocol and it is just reverse of ARP.

RARP perform following steps to obtain an IP address from the server.

1. The sender broadcast the RARP request to all the other host present in the network.
2. The RARP request packet contains the physical address of the sender.
3. All the host receiving the RARP request packet process it but, the authorized host only which can serve RARP service, responds to the RARP request packet such host are known as RARP Server.
4. The authorized RARP server replies directly to requesting host with the RARP response packet which contains IP address for the sender.



RARP is outdated now because of two reasons. First, the RARP is using the broadcast service of the data-link layer; that means the RARP must be present at each network. Second, RARP only provides IP address but today the computer also need other information.

3.3 Difference between ARP And RARP

Parameters	ARP	RARP
Abbreviation for	Address resolution protocol	Reverse Address Resolution Protocol
Broadcast MAC/IP	Nodes use ARP broadcast in LAN by using broadcast MAC address	RARP uses Broadcast IP address
Mapping	Maps IP address of node to its MAC Address	Maps 48 bit MAC address to IP address
Usage	Used by host or Router to find physical address of another host/Router in LAN.	Used by thin clients with limited facilities
Table maintained by	Local Host maintains ARP table	RARP Server maintains RARP table
Reply information	ARP reply is used to update ARP table	RARP reply is used to configure IP address in local host

SYBSCIT

Internet Technology

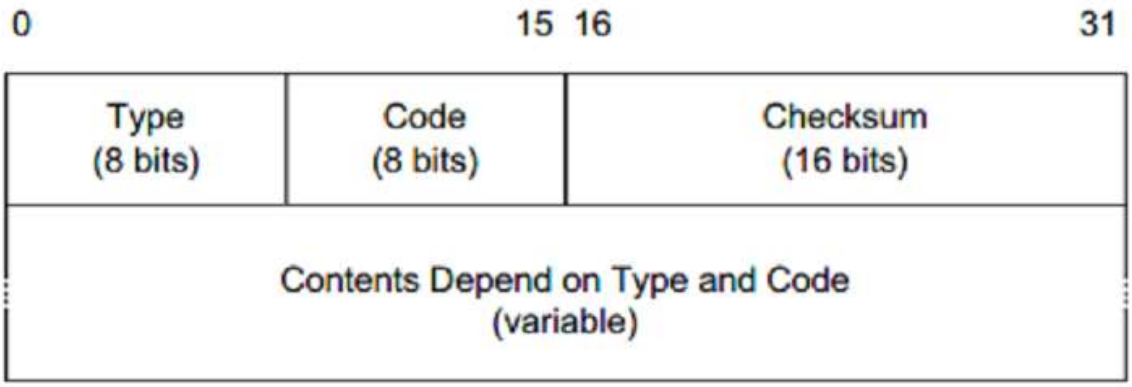
Chapter 4

ICMP4 & Mobile IP

4.1 ICMPv4
4.2 Mobile IP

4.1 ICMPv4

- ICMP (Internet Control Message Protocol) is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets.
- ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet that a router, service or host cannot be reached for packet delivery. Any IP network device has the capability to send, receive or process ICMP messages.
- Internet Control Message Protocol (ICMP) is used in conjunction with IP to provide diagnostics and control information related to the configuration of the IP protocol layer and the disposition of IP packets.
- ICMP provides for the delivery of error and control messages that may require attention.
- ICMP does not provide reliability for IP; it indicates certain classes of failures and configuration information.
- The most common cause of packet drops (buffer overrun at a router) does not elicit any ICMP information. Other protocols, such as TCP, handle such situations
- Because of the ability of ICMP to affect the operation of important system functions and obtain configuration information, hackers have used ICMP messages in a large number of attacks.
- As a result of concerns about such attacks, network administrators often arrange to block ICMP messages with firewalls, especially at border routers.
- If ICMP is blocked, however, a number of common diagnostic utilities (e.g., ping, traceroute) do not work properly.



In ICMPv4:

- 42 different values are reserved for the **Type** field [ICMPTYPES], which identify the particular message. Only about 8 of these are in regular use.
- Many types of ICMP messages also use different values of the **Code** field to further specify the meaning of the message.
- The **Checksum** field covers the entire ICMPv4 message; in ICMPv6 it also covers a pseudo-header derived from portions of the IPv6 header. The algorithm used for computing the checksum is the same as that used for the IP header checksum

- The **Internet Control Message Protocol** (**ICMP**) is an error reporting protocol that is an integral part of the IP protocol.
- ICMP communicate control data, information data, and error recovery data across the network. Problems that are less severe than transmission errors result in error conditions that can be reported.
- For example, suppose some of the physical paths in Internet fail causing the Internet to be partitioned into two sets of networks with no path between the sets. A datagram sent from a host in one set to a host in other cannot be delivered.
- The TCP/IP suite includes a protocol called ICMP that IP uses to send error messages when condition such as the one described above arises.
- The protocol is required for a standard implementation of IP. We will see that the two protocols are co-dependent.
- IP uses ICMP when it sends an error message, and ICMP uses IP to transport messages.

➢ Error messages defined by ICMP protocol:

1. **Source Quench**
- A router or host whose receive communication buffers are nearly full normally triggers this message.
- A source quench message is sent to the sending host, the receiver is simply requesting the sending host to reduce the rate at which it is transmitting until advised otherwise.

2. **Time Exceeded**
- A time-exceeded message is sent in two cases.

- Whenever a router reduces the TIL field in a data gram to zero. The router discards the datagram and sends a time exceeded message.
- In addition, a time exceeded message is sent by a host if the reassembly timer expires before all fragments from a given datagram arrive

3. **Route Redirect**

- A router sends this message to a host that is requesting its routing services.
- When a host creates a datagram destined for a network, the host sends the datagram to a router, which forwards the datagram to its destination.
- If a router determines that a host has incorrectly sent a datagram that should be sent to a different router, the router uses route redirect message to cause the host to change its route.
- In this manner, a route redirect message improves the efficiency of the routing process by informing the req4esting host of a shorter path to the desired destination.

4. **Host Unreachable**

- Whenever a gateway or a router determines that a datagram cannot be delivered to its final destination due to link failure or bandwidth congestion, an ICMP host unreachable message is sent to the originating node on the network.
- Normally, the message includes the reason the host cannot be reached.

5. **Fragmentation and Reassembly**

- The largest datagram the IP protocol can handle is 64 Kbytes.
- The maximum datagram size is dictated by the width of the total length field in the IP header. Realistically, most underlying data link technologies cannot accommodate this data size.
- For example, the maximum size of the data frame supported by Ethernet is 1,514 bytes.
- Whenever an upper-layer protocol delivers data segments whose sizes exceed the limit allowed by the underlying network, IP breaks the data into smaller pieces that are manageable within the allowed limit.
- The small data grams are then sent to the target host, which reassembles them for subsequent delivery to an upper-layer protocol.
- Although all data fragments are normally delivered using the same route, in some situations a few of them might traverse alternative routes.
- Fragments following different routes, however, stand the chance of reaching their-destination out of the order in which they were sent.
- To allow for recovery from such behaviour, IP employs the fragmentation-offset field in its header. The fragmentation-offset field includes sequencing information that the remote IP host uses to recover the sequence in which the data grams were sent.
- IP also uses the information in the fragmentation offset field to detect missing fragments
- Data is not passed to the protocol described in the protocol field unless all related fragments are duly received and reordered. This process of fragment recovery and re-sequencing is called data reassembly.
- A host that creates a datagram can set a bit in the flag field to specify the fragmentation. Among other bits, the flag field includes a more fragments bit, which is set to I in all fragments belonging to a datagram except for the final fragment.
- This ensures about the receiving of all fragments of a datagram.

1. **Echo request/Echo reply -**

- These two ICMP messages are exchanged between ICMP software on any two hosts in a bid to check connectivity between them.

- The ping command is an example of a diagnostic command commonly used by network users to check for the reach ability of a certain host.
- Whenever ping is invoked at the command line, ICMP echo request message is sent to the target host.
- If the target host is operational and connected to the network, it responds with an echo reply message as proof of reach ability. In other words, the reply carries the same data as the request.

2. **Address Mask Request/Reply**

- A host broadcasts an address mask request when it boots, and routers that receive the request send an address mask reply that contains the correct 2-bit subnet mask being used on the network.

4.2 Mobile IP

- In response to the increasing popularity of palmtop and other mobile computers, Mobile IP was developed to enable computers to maintain Internet connectivity while moving from one Internet attachment point to another.
- Although Mobile IP can work with wired connections, in which a computer is unplugged from one physical attachment point and plugged into another, it's particularly suited to wireless connections
- The term mobile in this context implies that a user is connected to one or more applications across the Internet, that the user's point of attachment changes dynamically, and that all connections are automatically maintained despite the change.
- This is in contrast to a user, such as a business traveler, with a portable computer of some sort, who arrives at a destination and uses the computer's notebook to dial into an ISP Internet service provider.
- In this latter case, the user's Internet connection is terminated each time the user moves, and a new connection is initiated when the user dials back in.
- Each time an Internet connection is established, software in the point of attachment (typically an ISP) is used to obtain a new, temporarily assigned IP address.
- For each application-level connection (such as FTP or web connection), this temporary IP address is used by the user's correspondent. A better term for this kind of use is nomadic.
- Mobile IP (or MIP) is an Internet Engineering Task Force (IETF) standard communications protocol that is designed to allow mobile device users to move from one network to another while maintaining a permanent IP address.
- It enables the transfer of information to and from mobile computers, such as laptops and wireless communications.
- The mobile computer can change its location to a foreign network and still access and communicate with and through the mobile computer’s home network.

➢ Enhance Mobility for Mobile IP

- Mobile IP – A technology which supports mobile data and applications that are dealing with wireless connectivity.

- A user may now disconnect his computer in the office and reconnect from another site within the same office or elsewhere.

- Mobile IP or IP-Mobility Management (IP-MM) is an open standard communication protocol defined by Internet Engineering Task Force that allows mobile device users to move from one network to another without changing their IP address as a change in the IP address will interrupt ongoing TCP/IP communications.

- Mobile IP is an enhancement of the Internet Protocol which allows a node to change its point of attachment to the Internet without needing to change its IP address.

- Mobile IP is independent of the physical layer technology as the mobility functions are performed at the network layer – any media that can support IP can support Mobile IP.

➢ Components of a Mobile IP

Mobile Node:

- A device such as a cell phone, personal digital assistant, or laptop whose software enables network roaming capabilities

The Foreign Agent:

- A router that may function as the point of attachment for the mobile node when it roams to a foreign network delivers packets from the home agent to the mobile node.

The Home Agent:

- A router on the home network serving as the anchor point for communication with the mobile node; its tunnel packets from a device on the Internet, called a correspondent node, to the roaming mobile node.

➢ Capabilities Of Mobile IP

Discovery:-

- A mobile node uses a discovery procedure to identify prospective home agents and foreign agents
- A mobile node first determines its connected location by using ICMP router discovery messages.
- If it's connected location is with the local network, then the normal IP routing is used for the communication.
- When a mobile node determines that it has moved to a foreign network it obtains a care-of address from the foreign agent reflecting its current location.

Registration:-

- A mobile node uses an authenticated registration procedure to inform its home agent of its care-of address.
- If the connected location is identified as foreign location, then the mobile node looks for a foreign agent and registers itself with the foreign location and the foreign agent, in turn, notifies the home agent and creates a tunnel between itself and the home agent.
- During this phase, the Mobile node sends a registration request message to the foreign agent which forwards the message to the home agent.
- The home agent sends back a reply after updating its registration table with the home address and "care-of" address mapping

Tunneling:-

- Tunneling is used to forward IP datagrams from a home address to a care-of address
- The method by which mobile IP receives information from a network is called tunneling.

SYBSCIT

Internet Technology

Chapter 5

Unicast Routing Protocols (RIP, OSPF and BGP)

5.1 Introduction to Unicast Routing Protocols

Introduction

- In unicast routing a packet needs to go from a single source to a single destination
- Unicast is the term used to describe communication where a piece of information is sent from one point to another point.
- In this case there is just one traffic, many streams of IP packets that move across networks flow from a single point, such as a website server, to a single endpoint such as a client PC.
- This is the most common form of information transference on networks. nder, and one receiver.
- Unicast transmission, in which a packet is sent from a single source to a specified destination, is still the predominant form of transmission on LANs and within the Internet.
- All LANs (e.g. Ethernet) and IP networks support the unicast transfer mode, and most users are familiar with the standard unicast applications (e.g. http, smtp, ftp and telnet) which employ the TCP transport protocol.
- Most of the traffic on the internet and intranets known as unicast data or unicast traffic is sent with specified destination. Routing unicast data over the internet is called unicast routing.
- It is the simplest form of routing because the destination is already known. Hence the router just has to look up the routing table and forward the packet to next hop.

5.1.1 Routing Algorithms

- A Routing Algorithm is a method for determining the routing of packets in a node.
- For each node of a network, the algorithm determines a routing table, which in each destination, matches an output line. The algorithm should lead to a consistent routing, that is to say without loop.
- This means that you should not route a packet a node to another node that could send back the package.
- Routing is the process of selecting paths in a network along which to send network traffic.
- Goals of routing are correctness, simplicity, Robustness, Stability, Fairness and Optimality.
- Routing is performed for many kinds of network, including the telephone network, electronic data networks and transportation networks.
- Routing Algorithms can be classified based on the following:

1. Static or Dynamic Routing

Static Routing

- A router with manually configured routing tables is known as a static router.
- A network administrator, with knowledge of the internetwork topology, manually builds and updates the routing table, programming all routes in the routing table.
- Static routers can work well for small internetworks but do not scale well to large or dynamically changing internetworks due to their manual administration.

Dynamic Routing

- A router with dynamically configured routing tables is known as a dynamic router.
- Dynamic routing consists of routing tables that are built and maintained automatically through an ongoing communication between routers.
- This communication is facilitated by a routing protocol, a series of periodic or on-demand messages containing routing information that is exchanged between routers.
- Except for their initial configuration, dynamic routers require little ongoing maintenance, and therefore can scale to larger internetworks.

Updating Algorithm

Receive: a response RIP message
1. Add one hop to the hop count for each advertised destination.
2. Repeat the following steps for each advertised destination:
1. If (destination not in the routing table)
1. Add the advertised information to the table.
2. Else If (next-hop field is the same)
1. Replace entry in the table with the advertised one.
2. Else If (advertised hop count smaller than one in the table)
1. Replace entry in the routing table.

3. Return.

2. Distributed or Centralized

1. Centralized Routing

- centralized routing protocols belong to the family of dynamic routing protocols. In a network that uses a centralized routing protocol, a central processing device running on a “central” node gathers information on each link in the network.
- Then, this processing device uses the gathered information to compute routing tables for all other nodes.
- These routing protocols make use of a centralized database located at the central node for these computations.
- In other words, the routing table is kept at a single “central” node, which should be consulted when other nodes need to make a routing decision

2. Decentralized Routing

- Distributed routing protocols also belong to the family of dynamic routing protocols.
- Under distributed routing protocol, each device in the network is responsible for making routing decisions.
- There are two types of dynamic, distributed protocols called isolated (nodes do not communicate) and non-isolated (nodes communicate with each other).

3. Single path or Multi path

4. Flat or Hierarchical

Flat Routing Protocols

- Flat routing protocols distribute information as needed to any router that can be reached or receive information.
- No effort is made to organize the network or its traffic, only to discover the best route hop by hop to a destination by any path.

Hierarchical Routing Protocols

- Hierarchical routing protocols often group routers together by function into a hierarchy.
- A heirarchical routing protocol allows an administrator to make best use of his fast powerful routers in the backbone, and the slower, lower-powered routers may be used for network access at the edge of the network.
- The access routers form the first tier of the hierarchy, and the backbone routers form the second tier.

5. Intra Domain or Inter Domain

Intra Domain

- An intra-domain routing protocol (like RIP, EIGRP, OSPF or IS-IS) focuses on distributing routes based on link quality (bandwidth, latency) or load balancing.
- With that, each router might know a specific route for all the subnets in your network.

Inter Domain

- Inter domain routing protocol basically does the same thing: gives your router information about how to reach other networks.
- However, it consider the number of domains traversed. Furthermore, with BGP you can customize routing choices based on administrator settings, rather than best-route approach. You can use it, as an example, to connect to two providers

5.2 Routing Information Protocol (RIP)

Routing Information Protocol (RIP) is a distance-vector routing protocol, which can be used in **small** networks.

- **Distance:** Distance from its destination, as there is no meters or kilometers parameters; in routing we use the metric which is the same as distance.
- **Vector:** The direction, in routing, this information will be the interface and the IP address of the router to send traffic to.

➢ How Routing Information Protocol (RIP) works

- RIP uses a distance vector algorithm to decide which path to put a packet on to get to its destination. Each RIP router maintains a routing table, which is a list of all the destinations the router knows how to reach. Each router broadcasts its entire routing table to its closest neighbors every 30 seconds (224.0.0.9 for RIPv2). In this context, *neighbors* are the other routers to which a router is connected directly -- that is, the other routers on the same network segments this router is on. The neighbors, in turn, pass the information on to their nearest neighbors, and so on, until all RIP hosts within the network have the same knowledge of routing paths. This shared knowledge is known as *convergence*.
- If a router receives an update on a route, and the new path is shorter, it will update its table entry with the length and next-hop address of the shorter path. If the new path is longer, it will wait through a "hold-down" period to see if later updates reflect the higher value as well. It will only update the table entry if the new, longer path has been determined to be stable.
- If a router crashes or a network connection is severed, the network discovers this because that router stops sending updates to its neighbors, or stops sending and receiving updates along the severed connection. If a given route in the routing table isn't updated across six successive update cycles (that is, for 180 seconds) a RIP router will drop that route and let the rest of the network know about the problem through its own periodic updates.

- RIP is a standardized Distance Vector protocol, designed for use on smaller networks. RIP was one of the first true Distance Vector routing protocols, and is supported on a wide variety of systems.

➢ RIP adheres to the following Distance Vector characteristics:

• RIP sends out periodic routing updates (every 30 seconds)

• RIP sends out the full routing table every periodic update

• RIP uses a form of distance as its metric (in this case, hop-count)

• RIP uses the Bellman-Ford Distance Vector algorithm to determine the best "path" to a particular destination

Other characteristics of RIP include:

- RIP supports IP and IPX routing.
- RIP utilizes UDP port 520
- RIP routes have an administrative distance of 120.
- RIP has a maximum hop-count of 15 hops.

- Any network that is 16 hops away or more is considered unreachable to RIP, thus the maximum diameter of the network is 15 hops. A metric of 16 hops in RIP is considered a poison route or infinity metric. If multiple paths exist to a particular destination, RIP will load balance between those paths (by default, up to 4) only if the metric (hopcount) is equal. RIP uses a round-robin system of load-balancing between equal metric routes, which can lead to pinhole congestion.

Features of RIP

- RIP uses a modified hop count as a way to determine network distance. Modified reflects the fact that network engineers can assign paths a higher cost. By default, if a router's neighbor owns a destination network and can deliver packets directly to the destination network without using any other routers, that route has one hop. In network management terminology, this is described as a cost of 1.
- RIP allows only 15 hops in a path. If a packet can't reach a destination in 15 hops, the destination is considered unreachable. Paths can be assigned a higher cost (as if they involved extra hops) if the enterprise wants to limit or discourage their use. For example, a satellite backup link might be assigned a cost of 10 to force traffic follow other routes when available.

➢ RIP Timers

Timers in RPI help regulate performance. They include:

- **Update timer**
  Frequency of routing updates. Every 30 seconds IP RIP sends a complete copy of its routing table, subject to split horizon. IPX RIP does this every 60 seconds.
- **Invalid timer**
  Absence of refreshed content in a routing update. RIP waits 180 seconds to mark a route as invalid and immediately puts it into hold down.
- **Hold-down timers and triggered updates**
  Assist with stability of routes in the Cisco environment. Hold downs ensure that regular update messages do not inappropriately cause a routing loop. The router doesn't act on non-superior new information for a certain period of time. RIP's hold-down time is 180 seconds.
- **Flush timer**
  RIP waits an additional 240 seconds after hold down before it actually removes the route from the table.
  Other stability features to assist with routing loops include poison reverse. A poison reverse is a way in which a gateway node tells its neighbor gateways that one of the gateways is no longer connected. To do this, the notifying gateway sets the number of hops to the unconnected gateway to a number that indicates *infinite*, which in layman's terms simply means 'You can't get there.' Since RIP allows up to 15 hops to another gateway, setting the hop count to 16 is the equivalent of "infinite."

➢ **The interior gateway protocols can be divided into two categories:**
1) Distance-vector routing protocol
2) Link-state routing protocol.

Types of Interior gateway protocols

Distance-vector routing protocol

They use the Bellman-Ford algorithm to calculate paths. In Distance-vector routing protocols each router does not posses information about the full network topology. It advertises its distances from other routers and receives similar advertisements from other routers. Using these routing advertisements each router populates its routing table. In the next advertisement cycle, a router advertises updated information from its routing table. This process continues until the routing tables of each router converge to stable values.

This set of protocols has the disadvantage of slow convergence, however, they are usually simple to handle and are well suited for use with small networks. Some examples of distance-vector routing protocols are:

1. **Routing Information Protocol (RIP)**
2. **Interior Gateway Routing Protocol (IGRP)**

Link-state routing protocol

In the case of Link-state routing protocols, each node possesses information about the complete network topology. Each node then independently calculates the best next hop from it for every possible destination in the network using local information of the topology. The collection of best next hops forms the routing table for the node.

This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbors. In a link-state protocol, the only information passed between the nodes is information used to construct the connectivity maps.

Example of Link-state routing protocols are:

1. **Open Shortest Path First (OSPF)**
2. **Intermediate system to intermediate system (IS-IS)**

Topics include the routing update process, RIP routing metrics, routing stability, and routing timers.

Routing Updates

RIP sends routing-update messages at regular intervals and when the network topology changes. When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route. The metric value for the path is increased by 1, and the sender is indicated as the next hop. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. After updating its routing table, the router immediately begins transmitting routing updates to inform other network routers of the change. These updates are sent independently of the regularly scheduled updates that RIP routers send.

RIP Routing Metric

RIP uses a single routing metric (hop count) to measure the distance between the source and a destination network. Each hop in a path from source to destination is assigned a hop count value, which is typically 1. When a router receives a routing update that contains a new or changed destination network entry, the router adds 1 to the metric value indicated in the update and enters the network in the routing table. The IP address of the sender is used as the next hop.

RIP Stability Features

RIP prevents routing loops from continuing indefinitely by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops in a path is 15. If a router receives a routing update that contains a new or changed entry, and if increasing the metric value by 1 causes the metric to be infinity (that is, 16), the network destination is considered unreachable. The downside of this stability feature is that it limits the maximum diameter of a RIP network to less than 16 hops.

RIP includes a number of other stability features that are common to many routing protocols. These features are designed to provide stability despite potentially rapid changes in a network's topology. For example, RIP implements the split horizon and holddown mechanisms to prevent incorrect routing information from being propagated.

RIP Timers

RIP uses numerous timers to regulate its performance. These include a routing-update timer, a route-timeout timer, and a route-flush timer. The routing-update timer clocks the interval between periodic routing updates. Generally, it is set to 30 seconds, with a small random amount of time added whenever the timer is reset. This is done to help prevent congestion, which could result from all routers simultaneously attempting to update their neighbors. Each routing table entry has a route-timeout timer associated with it. When the route-timeout timer expires, the route is marked invalid but is retained in the table until the route-flush timer expires.

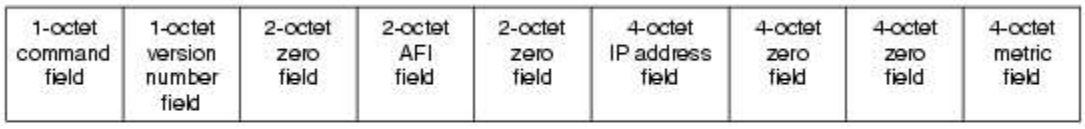
➢ RIP Packet Formats

The following section focuses on the IP RIP and IP RIP 2 packet formats illustrated in Figures 5.1 and 5.2 Each illustration is followed by descriptions of the fields illustrated.

- **RIP Packet Format**

Figure 5.1 illustrates the IP RIP packet format.

Figure 5.2 An IP RIP Packet Consists of Nine Fields



The following descriptions summarize the IP RIP packet format fields illustrated in Figure 5.1

• **Command**—Indicates whether the packet is a request or a response. The request asks that a router send all or part of its routing table. The response can be an unsolicited regular routing

update or a reply to a request. Responses contain routing table entries. Multiple RIP packets are used to convey information from large routing tables.

- **Version number**—Specifies the RIP version used. This field can signal different potentially incompatible versions.

- **Zero**—This field is not actually used by RFC 1058 RIP; it was added solely to provide backward compatibility with prestandard varieties of RIP. Its name comes from its defaulted value: zero.

- **Address-family identifier (AFI)**—Specifies the address family used. RIP is designed to carry routing information for several different protocols. Each entry has an address-family identifier to indicate the type of address being specified. The AFI for IP is 2.

- **Address**—Specifies the IP address for the entry.

- **Metric**—Indicates how many internetwork hops (routers) have been traversed in the trip to the destination. This value is between 1 and 15 for a valid route, or 16 for an unreachable route.

➢ RIP 2 Packet Format

The RIP 2 specification (described in RFC 1723) allows more information to be included in RIP packets and provides a simple authentication mechanism that is not supported by RIP. Figure 5.2 shows the IP RIP 2 packet format.

Figure 5.2 An IP RIP 2 Packet Consists of Fields Similar to Those of an IP RIP Packet

The following descriptions summarize the IP RIP 2 packet format fields illustrated in Figure 5.2

- **Command**—Indicates whether the packet is a request or a response. The request asks that a router send all or a part of its routing table. The response can be an unsolicited regular routing update or a reply to a request. Responses contain routing table entries. Multiple RIP packets are used to convey information from large routing tables.

- **Version**—Specifies the RIP version used. In a RIP packet implementing any of the RIP 2 fields or using authentication, this value is set to 2.

- **Unused**—Has a value set to zero.

- **Address-family identifier (AFI)**—Specifies the address family used. RIPv2's AFI field functions identically to RFC 1058 RIP's AFI field, with one exception: If the AFI for the first entry in the message is 0xFFFF, the remainder of the entry contains authentication information. Currently, the only authentication type is simple password.

- **Route tag**—Provides a method for distinguishing between internal routes (learned by RIP) and external routes (learned from other protocols).

- **IP address**—Specifies the IP address for the entry.

- **Subnet mask**—Contains the subnet mask for the entry. If this field is zero, no subnet mask has been specified for the entry.

- **Next hop**—Indicates the IP address of the next hop to which packets for the entry should be forwarded.

- **Metric**—Indicates how many internetwork hops (routers) have been traversed in the trip to the destination. This value is between 1 and 15 for a valid route, or 16 for an unreachable route.

➢ Limitations of the protocol

This protocol does not solve every possible routing problem. As mentioned above, it is primary intended for use as an IGP, in reasonably homogeneous networks of moderate size. In addition, the following specific limitations should be mentioned:

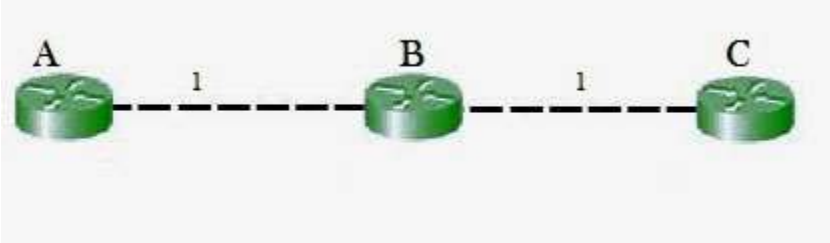
The protocol is limited to networks whose longest path involves 15 hops. The designers believe that the basic protocol design is inappropriate for larger networks. Note that this statement of the limit assumes that a cost of 1 is used for each network. This is the way RIP is normally configured. If the system administrator chooses to use larger costs, the upper bound of 15 can easily become a problem.

The protocol depends upon "counting to infinity" to resolve certain unusual situations. (This will be explained in the next section.) If the system of networks has several hundred networks, and a routing loop was formed involving all of them, the resolution of the loop would require either much time (if the frequency of routing updates were limited) or bandwidth (if updates were sent whenever changes were detected). Such a loop would consume a large amount of network bandwidth before the loop was corrected.
We believe that in realistic cases, this will not be a problem except on slow lines. Even then, the problem will be fairly unusual, since various precautions are taken that should prevent these problems in most cases.

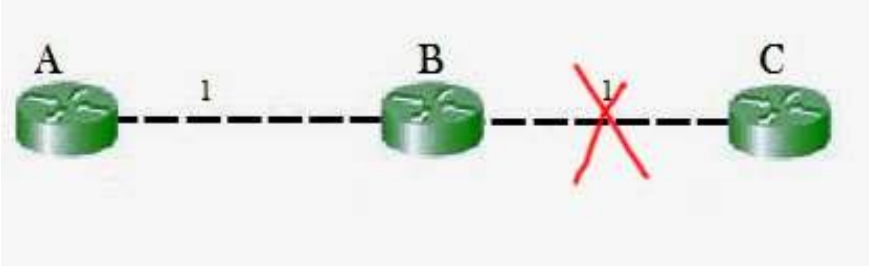
This protocol uses fixed "metrics" to compare alternative routes. It is not appropriate for situations where routes need to be chosen based on real-time parameters such a measured delay,

reliability, or load. The obvious extensions to allow metrics of this type are likely to introduce instabilities of a sort that the protocol is not designed to handle.

The main issue with **D**istance Vector **R**outing (DVR) protocols is Routing Loops, since Bellman-Ford Algorithm cannot prevent loops. This routing loop in DVR network causes Count to Infinity Problem. Routing loops usually occur when any interface goes down or two-routers send updates at the same time.
Counting to infinity problem:



So in this example, the Bellman-Ford algorithm will converge for each router, they will have entries for each other. B will know that it can get to C at a cost of 1, and A will know that it can get to C via B at a cost of 2.

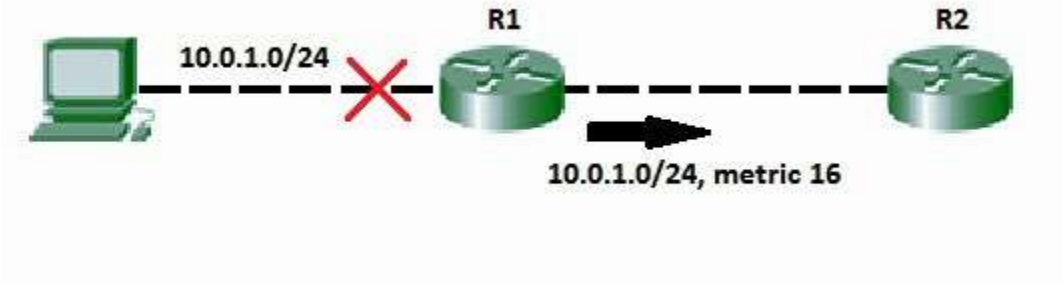


If the link between B and C is disconnected, then B will know that it can no longer get to C via that link and will remove it from it's table. Before it can send any updates it's possible that it will receive an update from A which will be advertising that it can get to C at a cost of 2. B can get to A at a cost of 1, so it will update a route to C via A at a cost of 3. A will then receive updates from B later and update its cost to 4. They will then go on feeding each other bad information toward infinity which is called as **Count to Infinity problem**.

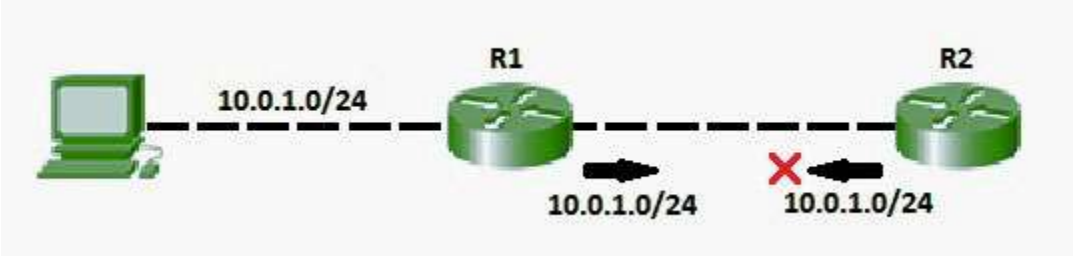
Solution for Count to Infinity problem:-

Route Poisoning:
When a route fails, distance vector protocols spread the *bad news* about a route failure by poisoning the route. Route poisoning refers to the practice of advertising a route, but with a special metric value called Infinity. Routers consider routes advertised with an infinite metric to have failed. Each distance vector routing protocol uses the concept of an actual metric value that represents infinity. RIP defines infinity as 16. The main disadvantage of poison reverse is that it can significantly increase the size of routing

announcements in certain fairly common network topologies.



Split horizon:
If the link between B and C goes down, and B had received a route from A , B could end up using that route via A. A would send the packet right back to B, creating a loop. But according to Split horizon Rule, Node A does not advertise its route for C (namely A to B to C) back to B. On the surface, this seems redundant since B will never route via node A because the route costs more than the direct route from B to C.
Consider the following network topology showing Split horizon-



- In addition to these, we can also use split horizon with route poisoning where above both technique will be used combinely to achieve efficiency and less increase the size of routing announcements.
- Split horizon with Poison reverse technique is used by Routing Information Protocol (RIP) to reduce routing loops. Additionally, **Holddown timers** can be used to avoid the formation of loops. Holddown timer immediately starts when the router is informed that attached link is down. Till this time, router ignores all updates of down route unless it receives an update from the router of that downed link. During the timer, If the down link is reachable again, routing table can be updated.

5.3 Open Shortest Path First

- Autonomous System Boundary Routers are responsible for delivering information about other autonomous systems into the current system.
- Within the autonomous system, the OSPF divides the hosts into areas.
- Each area has a area border router which summarizes information in the area and distributes it to other areas.
- All areas are connected to a special area called a backbone and areas can be connected to other areas. The backbone has the AS boundary router.

- . If the direct connection between an area and the backbone the administrator must create a virtual link which may pass though several other areas.

➢ OSPF Connections

- The point-to-point link connects two routers without any other host or router in between.
- The transient link is a network with several routers. Data can come in and out of the network through all the routers. All LANs and some WANs are of this type. To avoid a mesh topology, a router called a designated router is positioned at the center such that every other router has only one neighbor which is the designated router. The designated router is not an additional router. One of the existing routers is "designated" with the task.
- The stub link is a special case of the transient link where the number of routers equals one. All data going in and out of the network pass through this router.
- The virtual link is created when the direct link between two routers has been broken. Because it is not anymore possible for the data to go from router A to router B directly then the data would need to pass fro from router A to router C to router E then finally to router B for example.
- OSPF is a link-state routing protocol that calls for the sending of link-state advertisements (LSAs) to all other routers within the same hierarchical area. Information on attached interfaces, metrics used, and other variables is included in OSPF LSAs. As OSPF routers accumulate link-state information, they use the SPF algorithm to calculate the shortest path to each node.
- As a link-state routing protocol, OSPF contrasts with RIP and IGRP, which are distance-vector routing protocols. Routers running the distance-vector algorithm send all or a portion of their routing tables in routing-update messages to their neighbors.

➢ Link State Advertisement

Link state advertisements serve to inform the network about the information of an entity's neighbors. The type of LSA depends on the different entities that broadcast them.

1. Router LSA - the router announces its presence and lists the links to other routers or networks in the same area, together with the metrics to them. Type 1 LSAs are flooded across their own area only.

2. Network LSA - the designated router on a broadcast segment (e.g. Ethernet) lists which routers are joined together by the segment. Type 2 LSAs are flooded across their own area only.

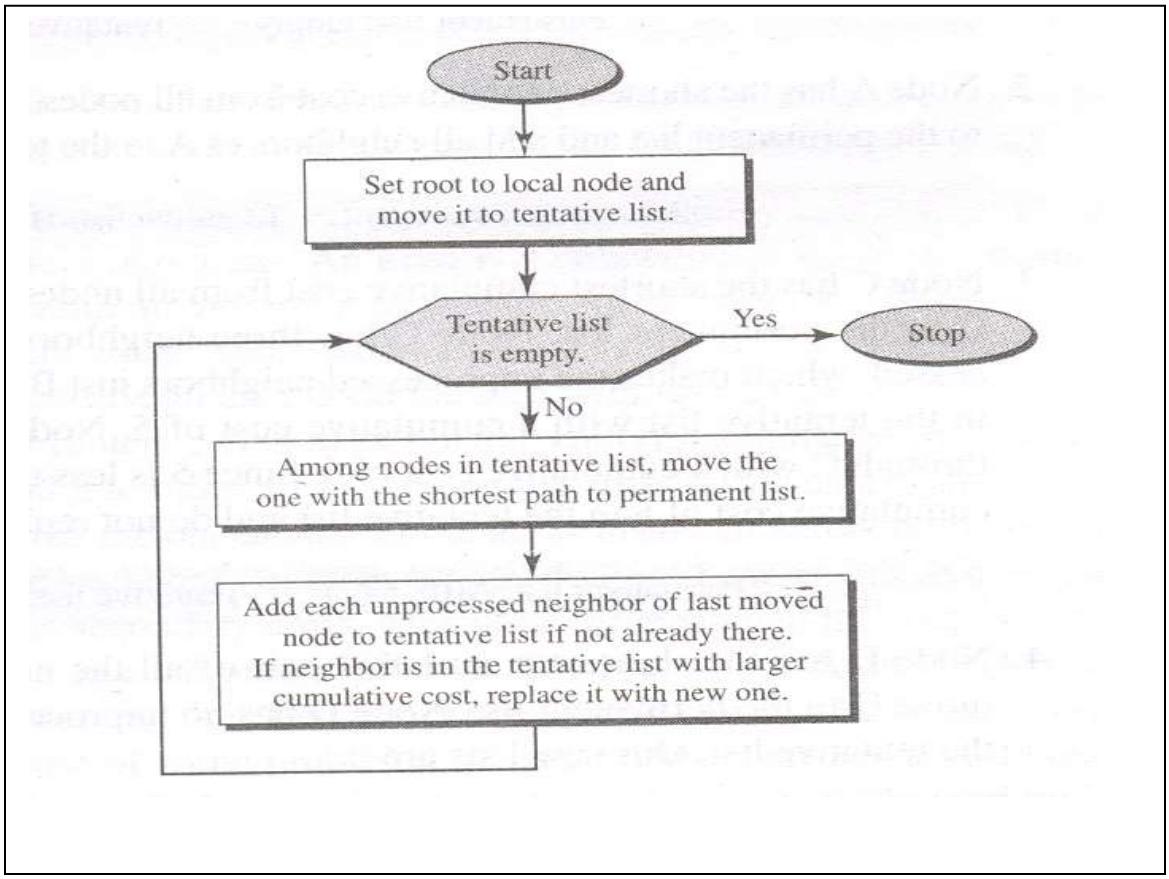
3. Summary LSA to Network - an Area Border Router (ABR) takes information it has learned on one of its attached areas and it can summarize it (but not by default) before sending it out on other areas it is connected to. This summarization helps provide scalability by removing detailed topology information for other areas, because their routing information is summarized into just an address prefix and metric.

4. ASBR-Summary LSA - this is needed because Type 5 External LSAs are flooded to all areas and the detailed next-hop information may not be available in those other areas. This is solved by an Area Border Router flooding the information for the router (i.e. the Autonomous System Boundary Router) where the type 5 originated.

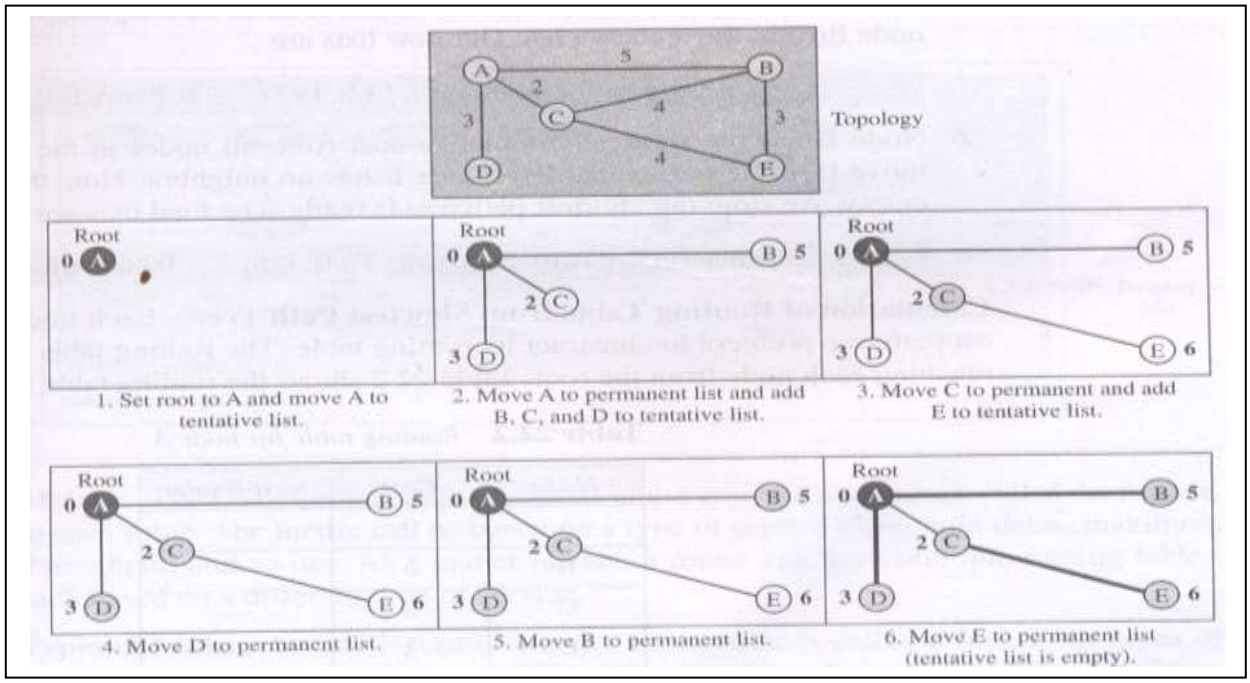
5. External LSA - these LSAs contain information imported into OSPF from other routing processes. They are flooded to all areas (except stub areas).

➢ The Dijkstra algorithm

- After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.
- A tree is a graph of nodes and links; one node is called the **root**. All other nodes can be reached from the root through only one single route. A **shortest path tree** is a tree in which the path between the root and every other node is the shortest. What we need for each node is a shortest path tree with that node as the root.
- The **Dijkstra algorithm** creates a **shortest path tree from a graph**. The algorithm divides the nodes into two sets: **tentative** and **permanent**. It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.
- We can define the algorithm by using the flowchart



Let us apply the algorithm to node A of our sample graph in Following Figure



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.

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(0)

2. Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to the permanent list and add all neighbors of A to the tentative list. Our new lists are

Permanent list:A(0) Tentative list:B(5),C(2)

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(0),C(2) Tentative list:B(5)

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(0),C(2),D(3) Tentative list:

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(0),C(2),D(3) Tentative list:

6. Node E has the shortest cumulative cost from all nodes in the tentative list. We move E to the permanent list. Node E has no neighbor. Now the tentative list is empty. We stop; our shortest path tree is ready. The final lists are

   Permanent list: A(0),C(2),D(3),E(6) Tentative list:

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 1 *Routing table for node A*

Node	*Cost*	*Next Router*
A	0	—

➢ Routing Hierarchy

Unlike RIP, OSPF can operate within a hierarchy. The largest entity within the hierarchy is the autonomous system (AS), which is a collection of networks under a common administration that share a common routing strategy. OSPF is an intra-AS (interior gateway) routing protocol, although it is capable of receiving routes from and sending routes to other ASs.

An AS can be divided into a number of areas, which are groups of contiguous networks and attached hosts. Routers with multiple interfaces can participate in multiple areas. These routers, which are called Area Border Routers, maintain separate topological databases for each area.

A topological database is essentially an overall picture of networks in relationship to routers. The topological database contains the collection of LSAs received from all routers in the same area. Because routers within the same area share the same information, they have identical topological databases.

The term *domain* sometimes is used to describe a portion of the network in which all routers have identical topological databases. Domain is frequently used interchangeably with AS.

An area's topology is invisible to entities outside the area. By keeping area topologies separate, OSPF passes less routing traffic than it would if the AS were not partitioned.

Area partitioning creates two different types of OSPF routing, depending on whether the source and the destination are in the same or different areas. Intra-area routing occurs when the source and destination are in the same area; interarea routing occurs when they are in different areas.

An OSPF backbone is responsible for distributing routing information between areas. It consists of all Area Border Routers, networks not wholly contained in any area, and their attached routers. Figure 5.3 shows an example of an internetwork with several areas.

In the figure 5.3, routers 4, 5, 6, 10, 11, and 12 make up the backbone. If Host H1 in Area 3 wants to send a packet to Host H2 in Area 2, the packet is sent to Router 13, which forwards the packet to Router 12, which sends the packet to Router 11. Router 11 then forwards the packet along the backbone to Area Border Router 10, which sends the packet through two intra-area routers (Router 9 and Router 7) to be forwarded to Host H2.

The backbone itself is an OSPF area, so all backbone routers use the same procedures and algorithms to maintain routing information within the backbone that any area router would. The backbone topology is invisible to all intra-area routers, as are individual area topologies to the backbone.

Areas can be defined in such a way that the backbone is not contiguous. In this case, backbone connectivity must be restored through virtual links. Virtual links are configured between any backbone routers that share a link to a nonbackbone area and function as if they were direct links.

Figure 5.3 An OSPF AS Consists of Multiple Areas Linked by Routers

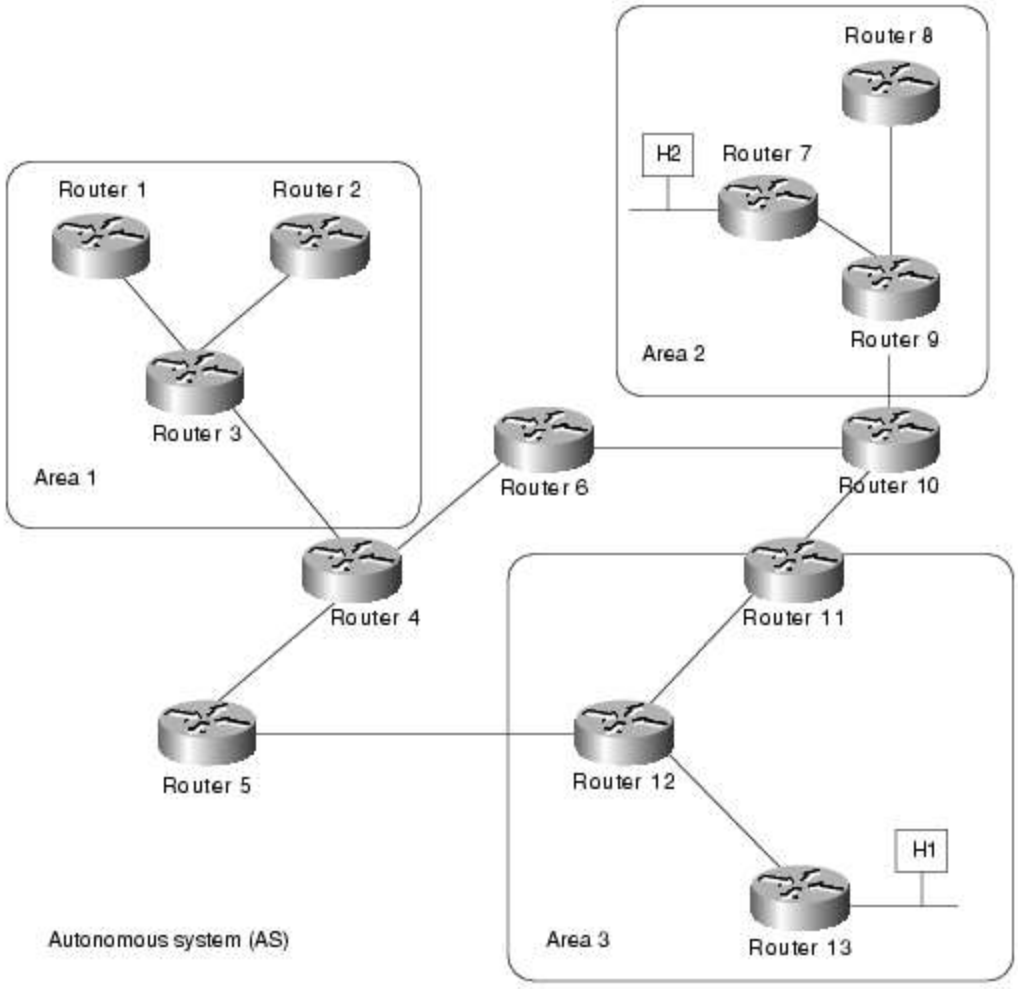


Figure 5.3 :- OSPF with Multiple Area Link

AS border routers running OSPF learn about exterior routes through exterior gateway protocols (EGPs), such as Exterior Gateway Protocol (EGP) or Border Gateway Protocol (BGP), or through configuration information. For more information about these protocols, see Chapter 39, "Border Gateway Protocol."

➢ SPF Algorithm

The *Shortest Path First (SPF)* routing algorithm is the basis for OSPF operations. When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional.

After a router is assured that its interfaces are functioning, it uses the OSPF Hello protocol to acquire neighbors, which are routers with interfaces to a common network. The router sends

hello packets to its neighbors and receives their hello packets. In addition to helping acquire neighbors, hello packets also act as keep alives to let routers know that other routers are still functional.

On multi access networks (networks supporting more than two routers), the Hello protocol elects a designated router and a backup designated router. Among other things, the designated router is responsible for generating LSAs for the entire multi access network. Designated routers allow a reduction in network traffic and in the size of the topological database.

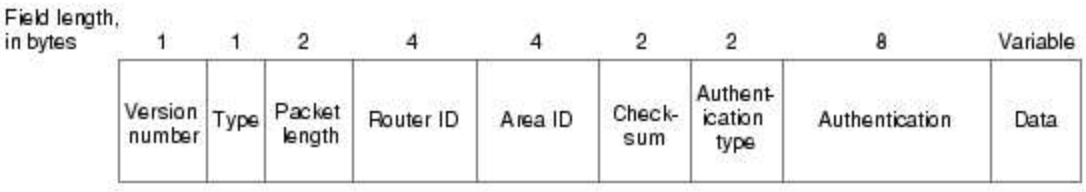
When the link-state databases of two neighboring routers are synchronized, the routers are said to be adjacent. On multi access networks, the designated router determines which routers should become adjacent. Topological databases are synchronized between pairs of adjacent routers. Adjacencies control the distribution of routing-protocol packets, which are sent and received only on adjacencies.

Each router periodically sends an LSA to provide information on a router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly, and the network's topology can be altered appropriately. From the topological database generated from LSAs, each router calculates a shortest-path tree, with itself as root. The shortest-path tree, in turn, yields a routing table.

➢ *OSPF Packet Format*

All OSPF packets begin with a 24-byte header, as illustrated in Figure 5.4.

Figure 5.4 OSPF Packets Consist of Nine Fields



The following descriptions summarize the header fields illustrated in Figure 46-2.

- **Version number**—Identifies the OSPF version used.

- **Type**—Identifies the OSPF packet type as one of the following:

– **Hello**—Establishes and maintains neighbor relationships.

– **Database description**—Describes the contents of the topological database. These messages are exchanged when an adjacency is initialized.

– **Link-state request**—Requests pieces of the topological database from neighbor routers. These messages are exchanged after a router discovers (by examining database-description packets) that parts of its topological database are outdated.

– **Link-state update**—Responds to a link-state request packet. These messages also are used for the regular dispersal of LSAs. Several LSAs can be included within a single link-state update packet.

– **Link-state acknowledgment**—Acknowledges link-state update packets.

- **Packet length**—Specifies the packet length, including the OSPF header, in bytes.

- **Router ID**—Identifies the source of the packet.

- **Area ID**—Identifies the area to which the packet belongs. All OSPF packets are associated with a single area.

- **Checksum**—Checks the entire packet contents for any damage suffered in transit.

- **Authentication type**—Contains the authentication type. All OSPF protocol exchanges are authenticated. The authentication type is configurable on per-area basis.

- **Authentication**—Contains authentication information.

- **Data**—Contains encapsulated upper-layer information.

➢ OSPF Features

Additional OSPF features include equal-cost, multipath routing, and routing based on upper-layer type-of-service (TOS) requests. TOS-based routing supports those upper-layer protocols that can specify particular types of service. An application, for example, might specify that certain data is urgent. If OSPF has high-priority links at its disposal, these can be used to transport the urgent datagram.

OSPF supports one or more metrics. If only one metric is used, it is considered to be arbitrary, and TOS is not supported. If more than one metric is used, TOS is optionally supported through the use of a separate metric (and, therefore, a separate routing table) for each of the eight combinations created by the three IP TOS bits (the delay, throughput, and reliability bits). For example, if the IP TOS bits specify low delay, low throughput, and high reliability, OSPF calculates routes to all destinations based on this TOS designation.

IP subnet masks are included with each advertised destination, enabling variable-length subnet masks. With variable-length subnet masks, an IP network can be broken into many subnets of various sizes. This provides network administrators with extra network-configuration flexibility.

5.4 BGP

The Border Gateway Protocol (BGP): The Border Gateway Protocol (BGP) is the core routing protocol of the Internet. It works by maintaining a table of IP networks or 'prefixes' which designate network reachability among autonomous systems (AS). It is described as a path vector protocol. BGP does not use traditional IGP metrics, but makes routing decisions based on path, network policies and/or rule sets. From January 2006, the current version of BGP, version 4, is codified in RFC 4271.

BGP supports Classless Inter-Domain Routing and uses route aggregation to decrease the size of routing tables. Since 1994, version four of the protocol has been in use on the Internet. All previous versions are now obsolete.

BGP was created to replace the EGP routing protocol to allow fully decentralized routing in order to allow the removal of the NSFNet Internet backbone network. This allowed the Internet to become a truly decentralized system.

Very large private IP networks can also make use of BGP. An example would be the joining of a number of large Open Shortest Path First (OSPF) networks where OSPF by itself would not scale to size. Another reason to use BGP would be multihoming a network for better redundancy.

Most Internet users do not use BGP directly. However, since most Internet service providers must use BGP to establish routing between one another (especially if they are multihomed), it is one of the most important protocols of the Internet. Compare this with Signaling System #7, which is the inter-provider core call setup protocol on the PSTN.

➢ BGP operation

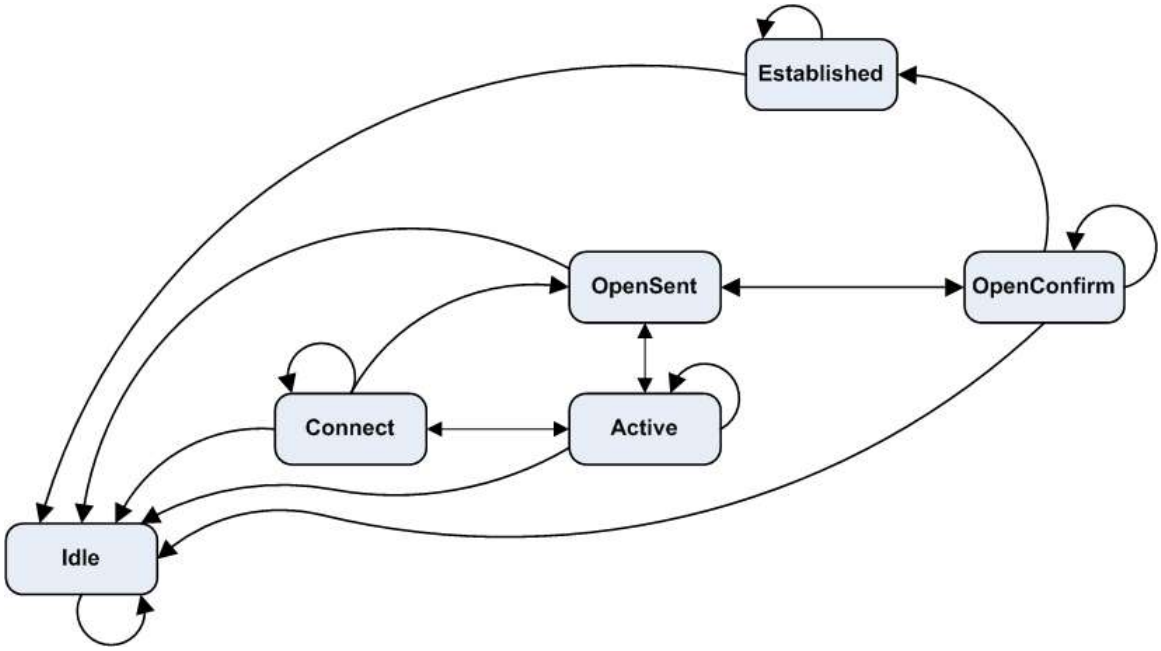
BGP neighbors, or peers, are established by manual configuration between routers to create a TCP session on port 179. A BGP speaker will periodically send 19-byte keep-alive messages to maintain the connection (every 60 seconds by default). Among routing protocols, BGP is unique in using TCP as its transport protocol.

When BGP is running inside an autonomous system (AS), it is referred to as Internal BGP (IBGP Interior Border Gateway Protocol). IBGP routes have an administrative distance of 200. When BGP runs between ASs, it is called External BGP (EBGP Exterior Border Gateway Protocol), and it has an administrative distance of 20. A BGP router that routes IBGP traffic is called a transit router. Routers that sit on the boundary of an AS and that use EBGP to exchange information with the ISP are border or edge routers.
In the simplest arrangement all routers within a single AS and participating in BGP routing must be configured in a full mesh: each router must be configured as peer to every other router. This causes scaling problems, since the number of required connections grows quadratically with the number of routers involved. To get around this, two solutions are built into BGP: route reflectors (RFC 2796) and confederations (RFC 3065).
Route reflectors reduce the number of connections required in an AS. A single router (or two for redundancy) can be made a route reflector: other routers in the AS need only be configured as peers to them.

Confederations are used in very large networks where a large AS can be configured to encompass smaller more manageable internal ASs. Confederations can be used in conjunction with route reflectors.

➢ Finite state machine

In order to make decisions in its operations with other BGP peers, a BGP peer uses a simple finite state machine that consists of six states: Idle, Connect, Active, OpenSent, OpenConfirm, and Established. For each peer-to-peer session, a BGP implementation maintains a state variable that tracks which of these six states the session is in. The BGP definition defines the messages that each peer should exchange in order to change the session from one state to another.



➢ BGP Header/Format

The size of BGP message format is 32 bit long. They encode with different type of message from the five types of messages function used to establish, maintain and update the neighbour relationship, notify and formatting errors about the peer router of BGP. All of these message have a common header that is sent to there BGP neighbour or also called a BGP peers, below is a example of the most common fields that will be found in a BGP message header:

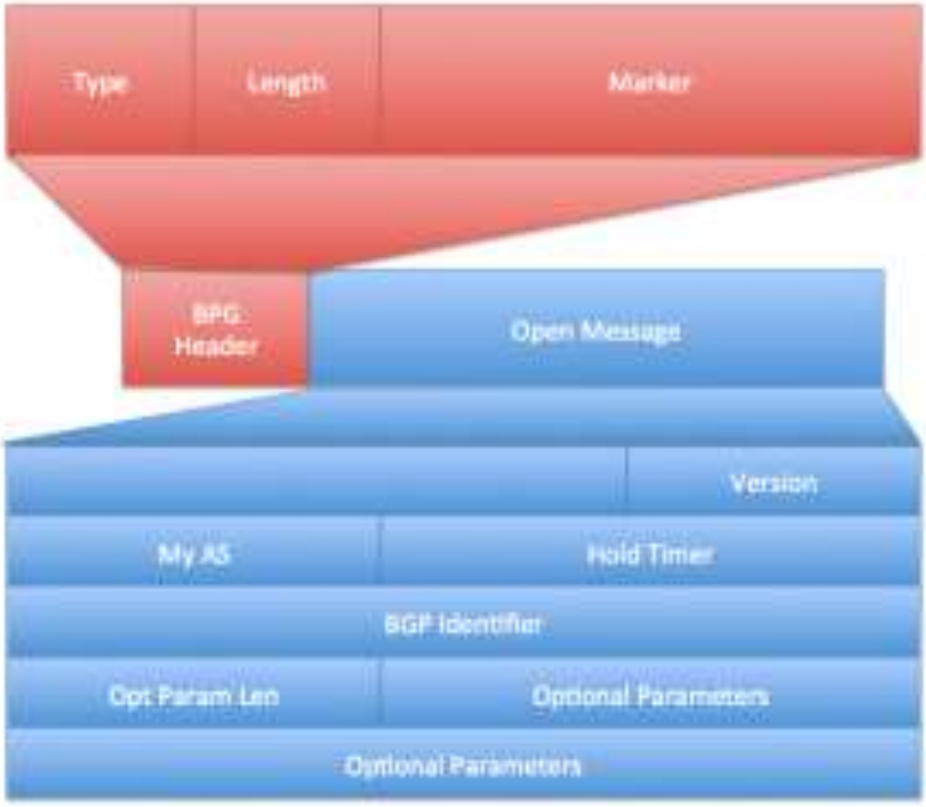


Fig 5.5: BGP Header

The fields in the BGP Message header are explain below:

- **Type:** This field is 8-bit long and it states what type of packet this is and what type of information it contained in the packet. There are five different type of message code defined by the Internet Engineering Task Force to standardlise BPG so that all vendors can them. The five type of message type codes are: 1-Open, 2-Update, 3-Notification, 4-Keepalive and the last and the only one code message type defined in RFC2918 is 5-Route-Refresh whereas the others are defined in the RFC 1771.

- **Length:** This filed message on the header is normally 2-byes long. This shows the total length of the BPG Message including the BGP headers. The value will always be between 19 and the maximum message allowed in any BGP message is 4096 byes.

- **Marker:** The remaining 16-bytes of the BGP Header is the Marker field this used to authentication BGP. If no authentication information is assigned or contained in this field then the marker is set to all ones or if the message is an open message.

➢ The Open Message

One of the five type of message that BGP send first is a Open message. A open message is sent after a TCP connection has been established with it peer router. Once a Open message has been send and been accept by the neighboring router a keep

alive message is sent to acknowledge the open message, after the router that sent a open message receives a keep alive message in return the BGP connect is in the established state and then update, keep alive and notification messages are to sent. Below is a open message format again the fields can be up to 32 bits message.

REPORT THIS AD

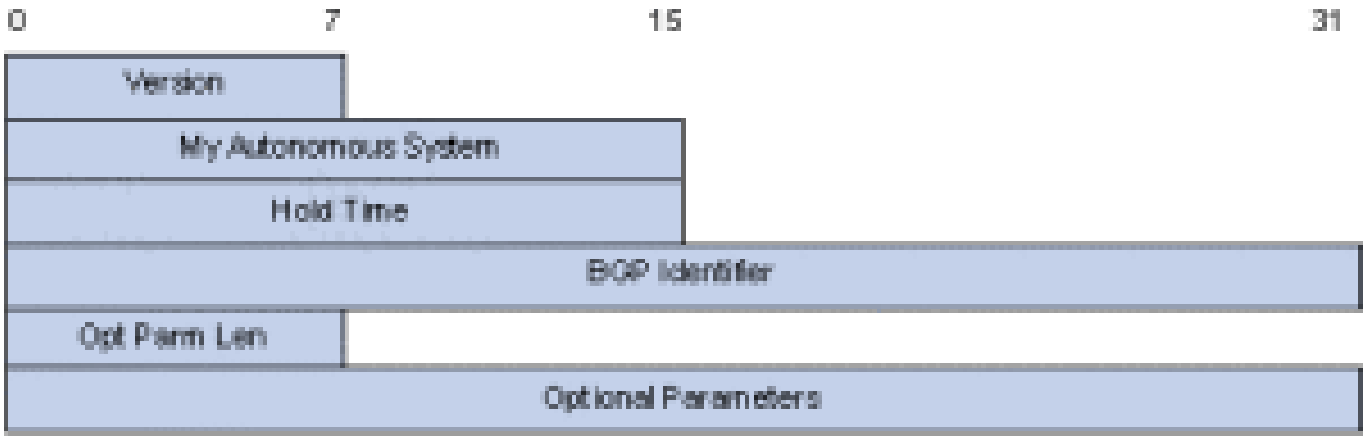


Fig 5.6: Open Message Format:

The BGP Open message contain more filed compare to the BGP header, below is a explanation of all the files in the BGP Open Message:

- Version: This 8 bit field contain the BGP Version that the originator is running. The most common BGP version used around the word is BGP-4
- My Autonomous System: This is a 2 byte field which content one of the most important message, which is the AS number of the originator router of the packet.
- Hold Time: This field is 16 bits long and it states the number of seconds the sender purposes for the hold time. The receiving peer will compare the value in the hold time field with it own configuration setting of hold time and give the value is same or smaller value it will accept the connection or reject the connection if any other difference. The hold time value must be within 0 or at least 3 seconds.
- BGP Identifier: This field can contain up to 32 bit value. The BGP Identifier is the routers ID of the senders peer. In Cisco vendors the highest IP address of the loopback interface on the router is BGP Identifier, if not loopback interface are configured than any highest IP address physical interface is selected as the value for the BGP Identifier.
- Optional Parameters Length: This field could be used in the future on new BGP features.

➢ The Update Message

The Update Message that gets sent to all the BGP router's including the peer, contains one of the most important information in any BGP message. The Update Message is responsible for exchanging routing information and possible route path to other networks between BGP neighbors.

REPORT THIS AD

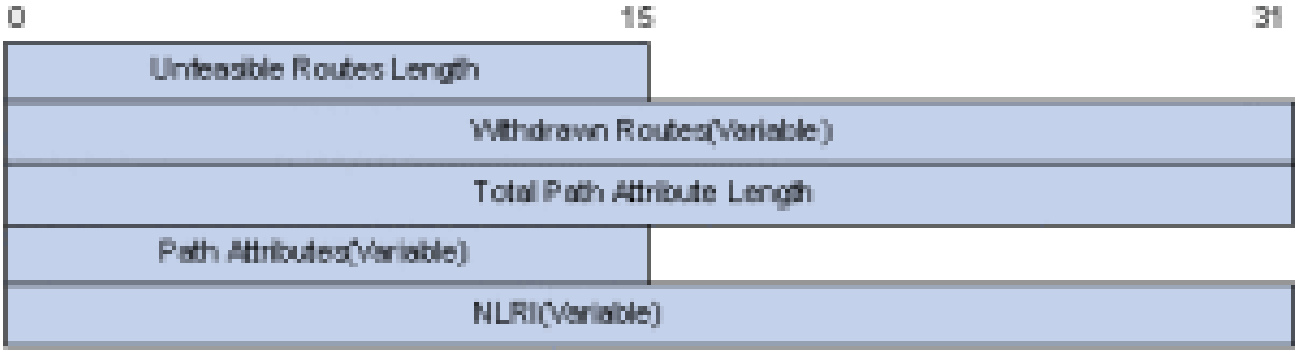


Fig 5.7: Update Message Header.

Below is a description of the field that are contained with the Update Message:

- Unfeasible Routes Length: This field in the message is a 2 Octets, this field indicated the receiving BGP router the total number of Withdrawn Routes has been withdrawn for the BGP route. When the length is 0 it means that no routes has been withdrawn from the BGP exchange information.
- Withdrawn Routes: This field in the message can be variable length in the message. In this part of the packet is the list of all the IP address of all the route/ip address that need to be withdrawn from the receiving BGP router.
- Total Path Attribute Length: In this part of the packet, the total number of length of the path attributes field in the byte. If the value is set to 0 then that will mean Network Layer Reachability information field is present in this packet
- Path Attributes: In this field of the packet, will be help the list of path attribute that are related to the Network Layer Reachability information. In this section each path attribute has the following: attribute type, attribute length, attribute value.
- Network Layer Reachability Information: This part of the packet contains a list of IP prefixes that can reach via different paths. "A Length value of 0 indicates a prefix that matches all IP prefixes."

➢ The Keepalive Message

"The Keepalive Message is sent between the BGP neighbors to maintain the connectivity. In this packet only three type of fields are used" [2]:

- AFI: Address Family Identifier.
- Res: Reserved. Set to 0.
- SAFI: Subsequent Address Family Identifier.

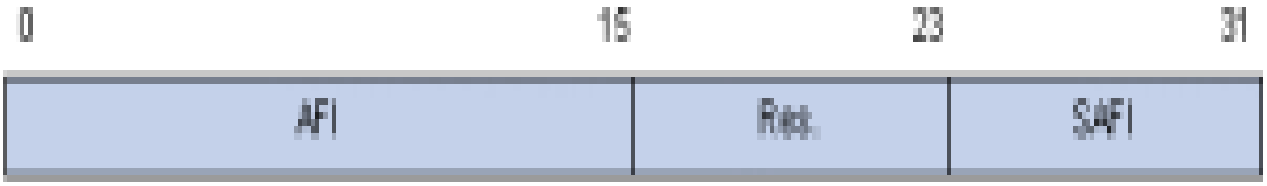


Fig 5.8: Keepalive Message format.

➢ The Notification Message

Notification message is used when an error is detected than BGP will send a notification message and then the BGP connection will be closed immediately after sending. This type of packet message only contains three different type of information, which is:

- Error Code: The error code of the notification
- Error Sub-code: Specifics the information about the nature of the problem that it has reported.
- Data: This contains depend upon the error code and the error sub code, it is normally used to diagnose the reason of the notification.

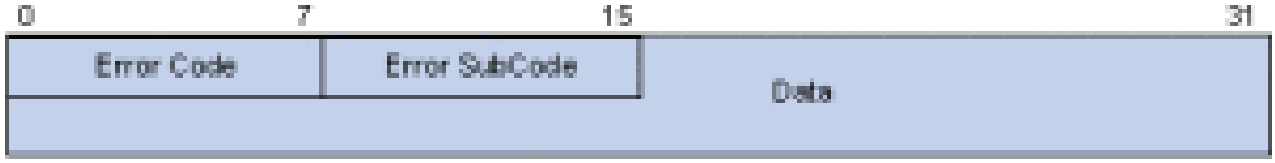


Fig 5.9: Notification Message format

Unit III

Transport Layer –

OBJECTIVE

We have several objectives for this chapter:

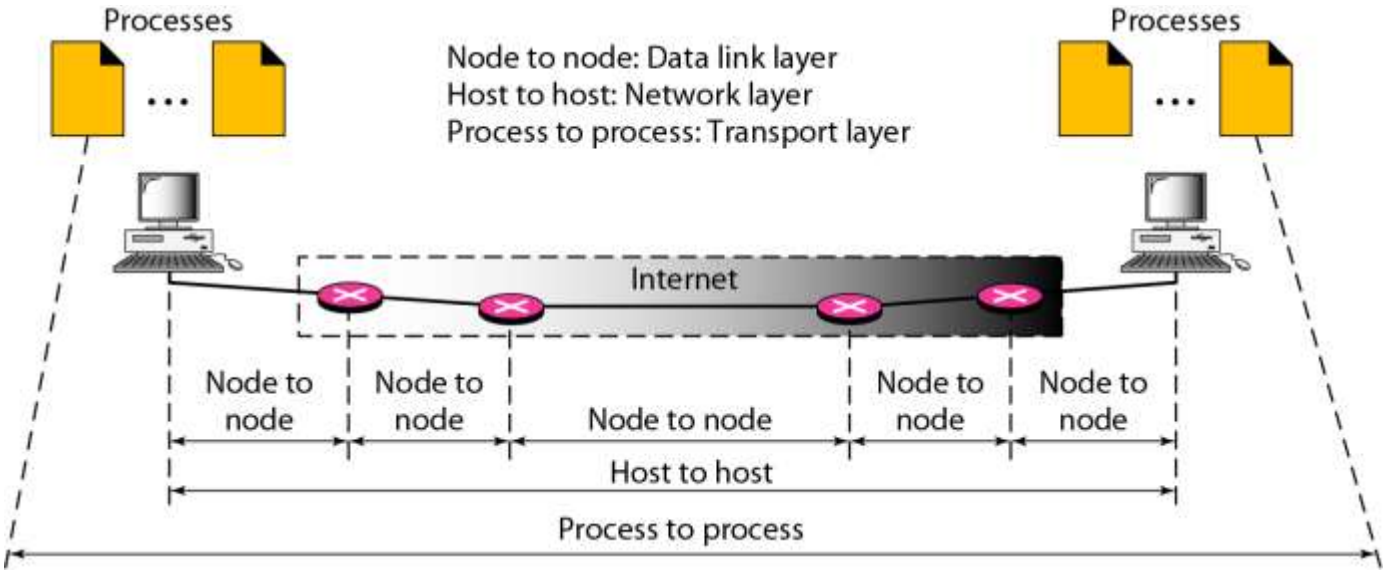
- ❑ To define process-to-process communication at the transport layer and compare it with host-to-host communication at the network layer.
- ❑ To discuss the addressing mechanism at the transport layer, to discuss port numbers, and to define the range port numbers used for different purposes.
- ❑ To explain the packetizing issue at the transport layer: encapsulation and decapsulation of messages.
- ❑ To discuss multiplexing (many-to-one) and demultiplexing (one-tomany) services provided by the transport layer.
- ❑ To discuss flow control and how it can be achieved at the transport layer.
- ❑ To discuss error control and how it can be achieved at the transport layer.
- ❑ To discuss congestion control and how it can be achieved at the transport layer.
- ❑ To discuss the connectionless and connection-oriented services at the transport layer and show their implementation using an FSM.
- ❑ To discuss the behavior of four generic transport-layer protocols and their applications: simple protocol, Stop-and-Wait protocol, Go-Back- *N* protocol, and Selective-Repeat protocol.
- ❑ To describe the idea of bidirectional communication at the transport layer using the piggybacking method.

TRANSPORT-LAYER SERVICES

The transport layer is located between the network layer and the application layer.
The transport layer is responsible for providing services to the application layer; it receives services from the network layer.
In this section, we discuss the services that can be provided by a transport layer

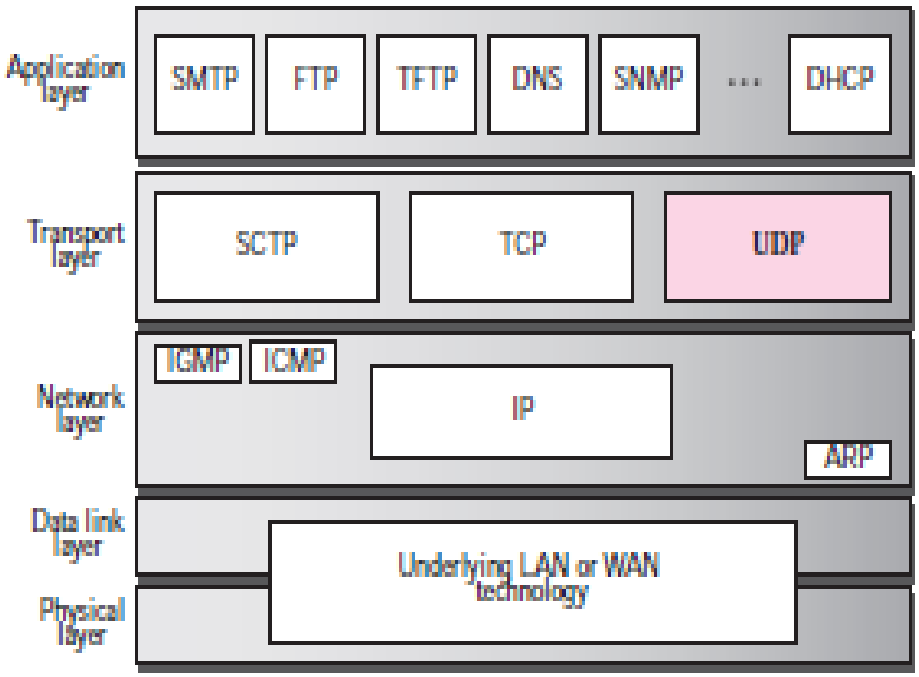
PROCESS-TO-PROCESS DELIVERY

• 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.

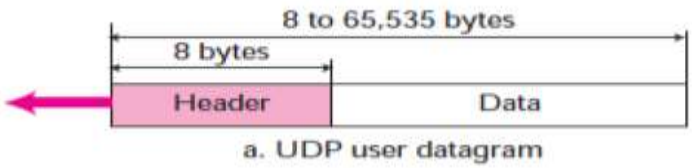
• Also, it performs very limited error checking.

• 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

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 –

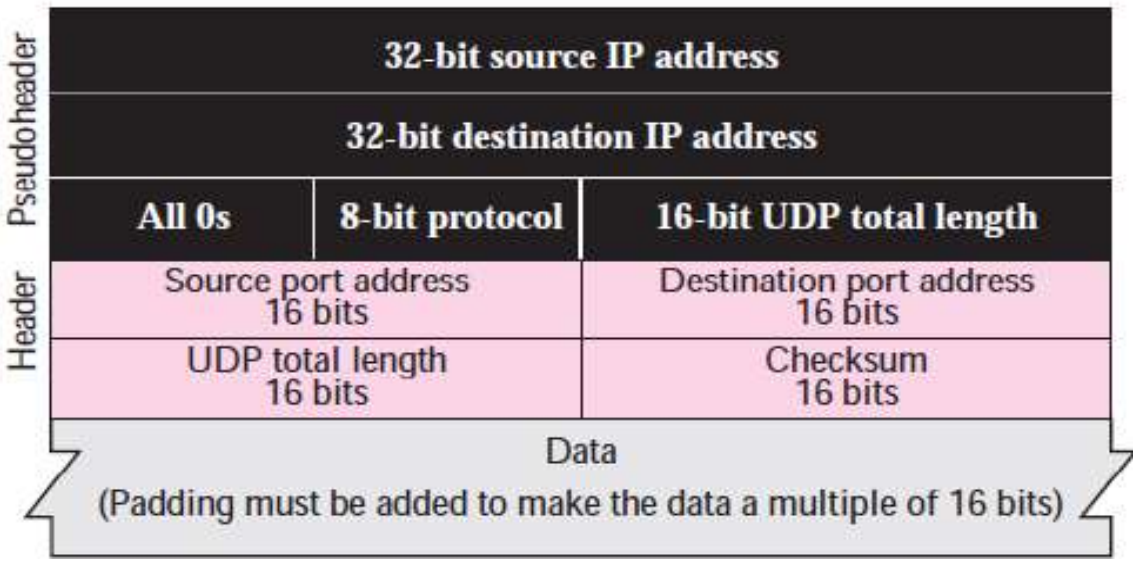
• UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.

• 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.

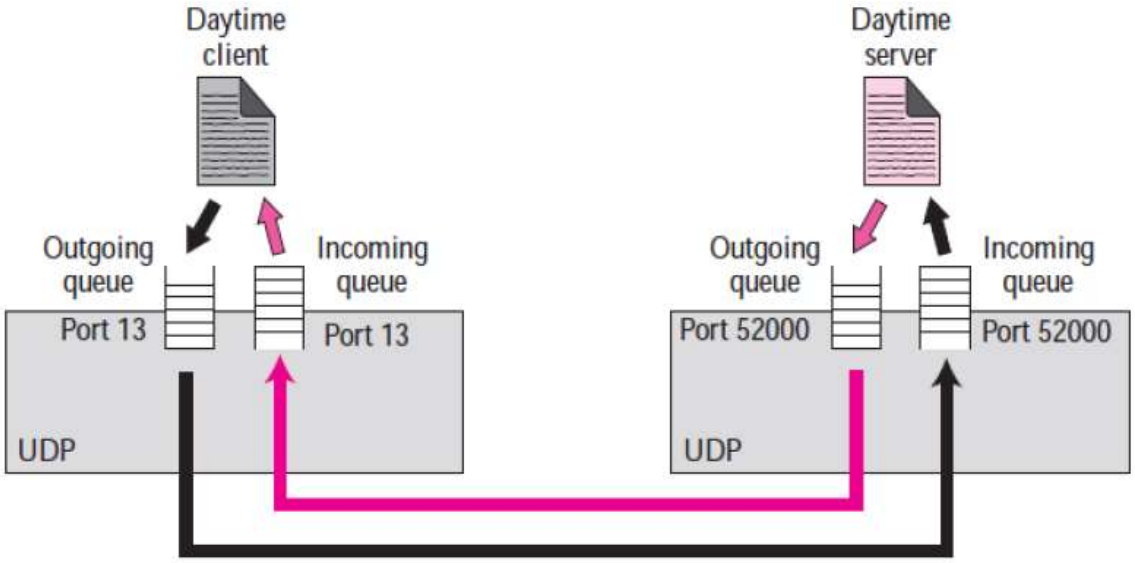
Encapsulation and Decapsulation –

• 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).



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.

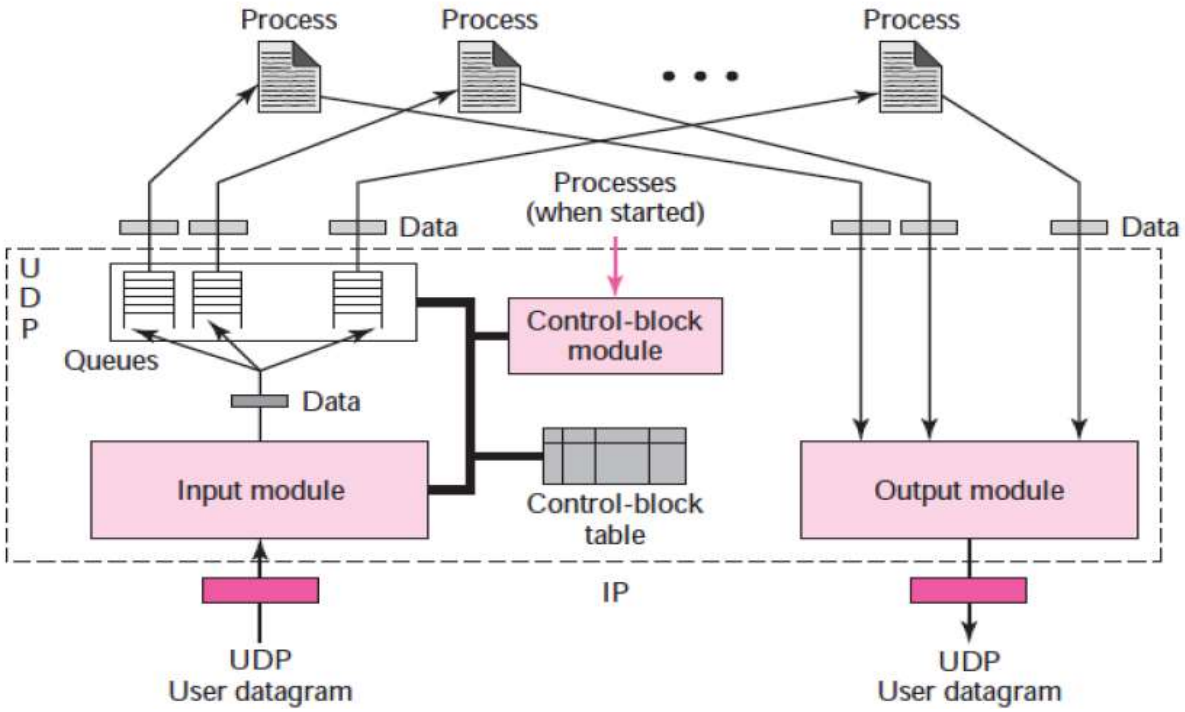
Lack of Error Control –
• UDP does not provide error control; it provides an unreliable service. Most applications expect reliable service from a transport-layer protocol. Although a reliable service is desirable, it may have some side effects that are not acceptable to some applications.
• When a transport 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.

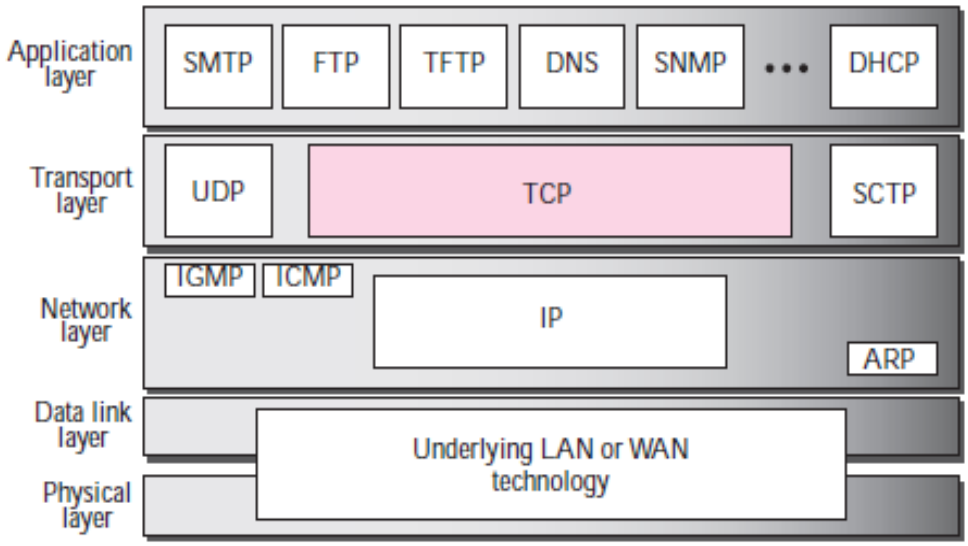
If the entry is found, the module uses the information in the entry to enqueue the data.

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



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.

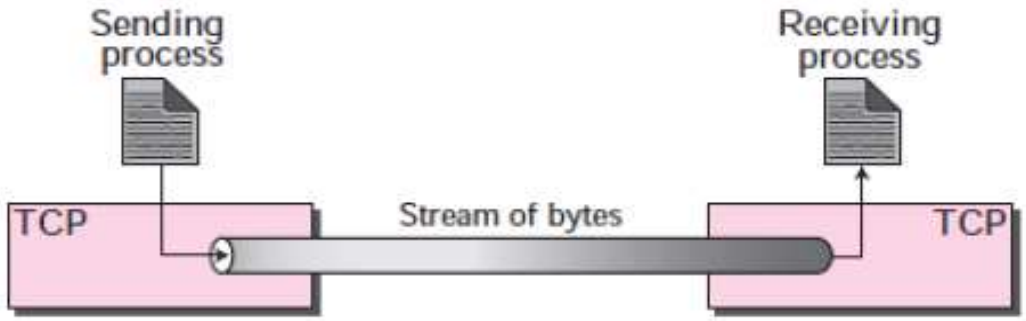
Port's of TCP

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

Port	*Protocol*	*Description*
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

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.

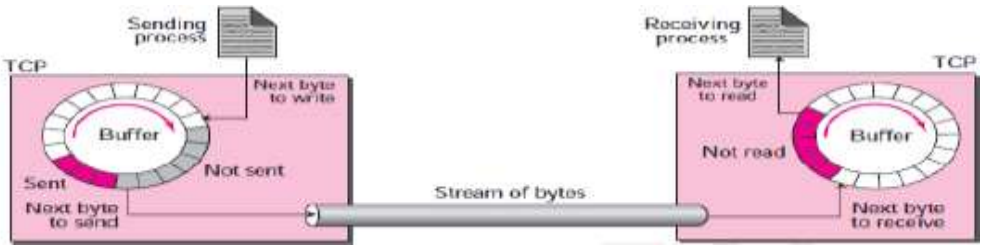
TCP creates an environment in which the two processes seem to be connected by an imaginary "tube— that carries their data across the Internet.



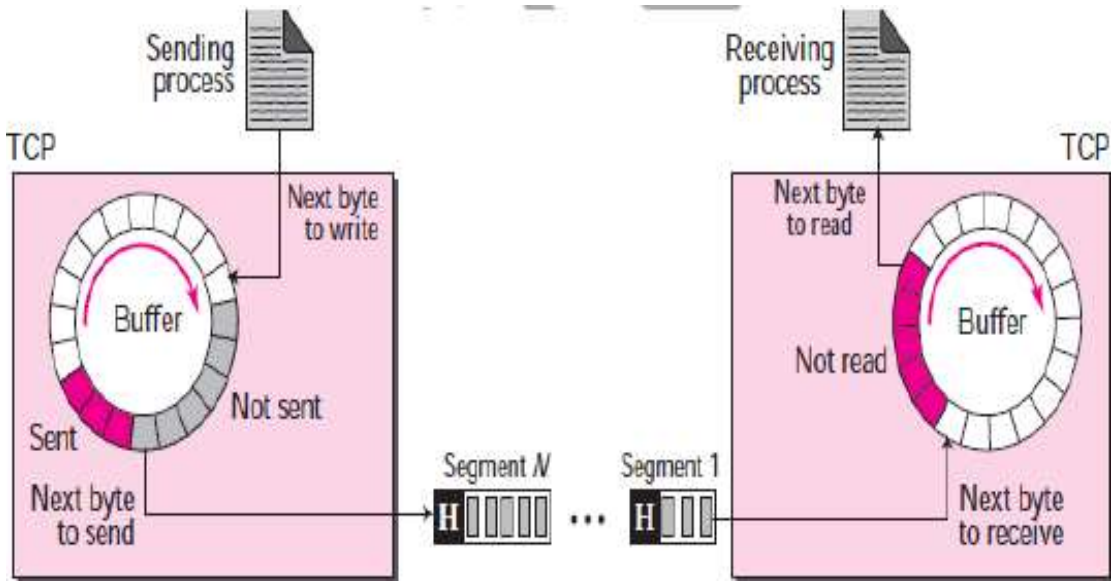
Sending and Receiving Buffers –

• 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.

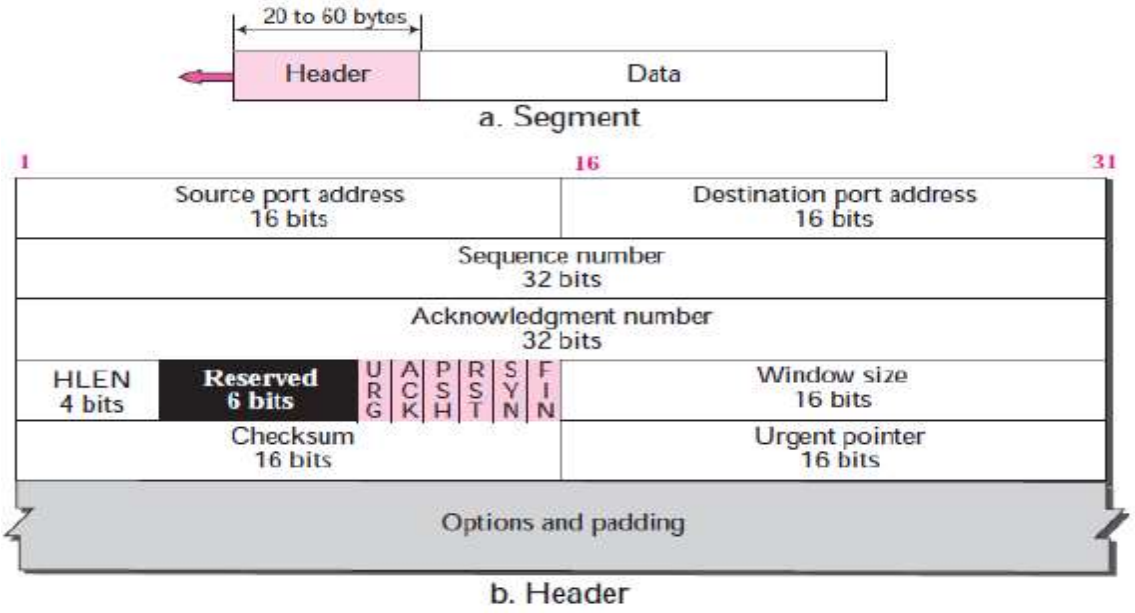
Sequence Number – After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

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. 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.

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* + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.

Header length –

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

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.

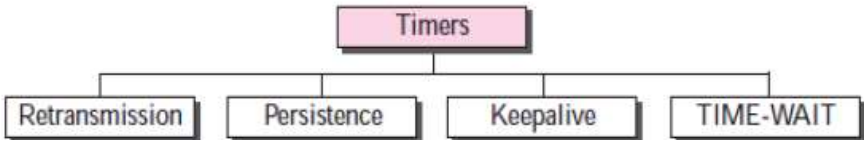
Window size –
• This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined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

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.

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.

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.

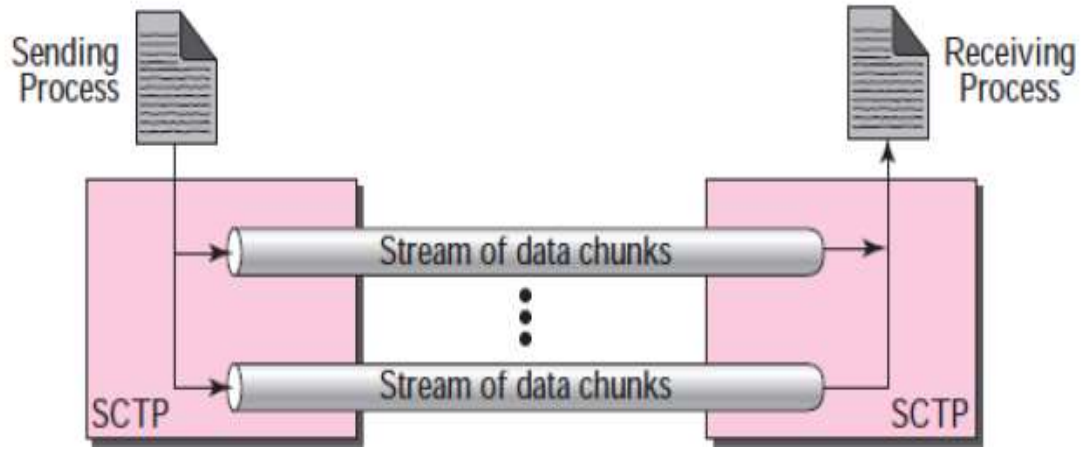
SCTP Services – *Process-to-Process Communication* – SCTP uses all well-known ports in the TCP space. Table lists some extra port numbers used by SCTP

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.

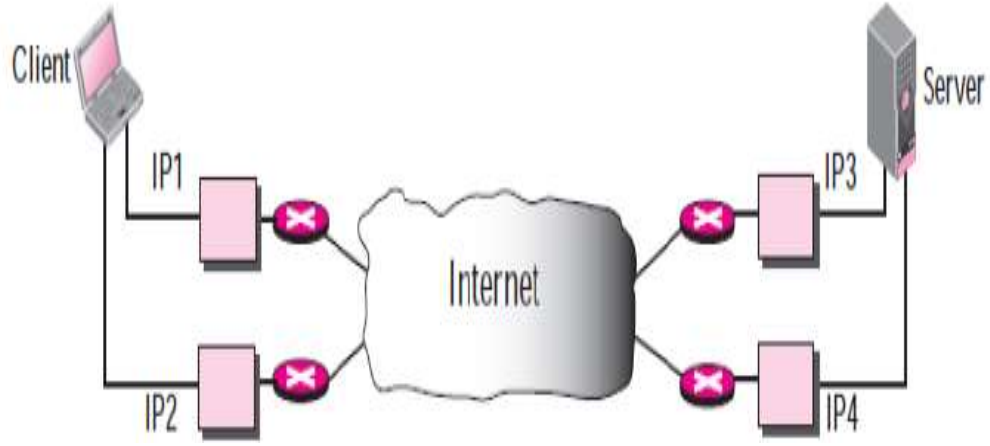


Multihoming –

• 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 –

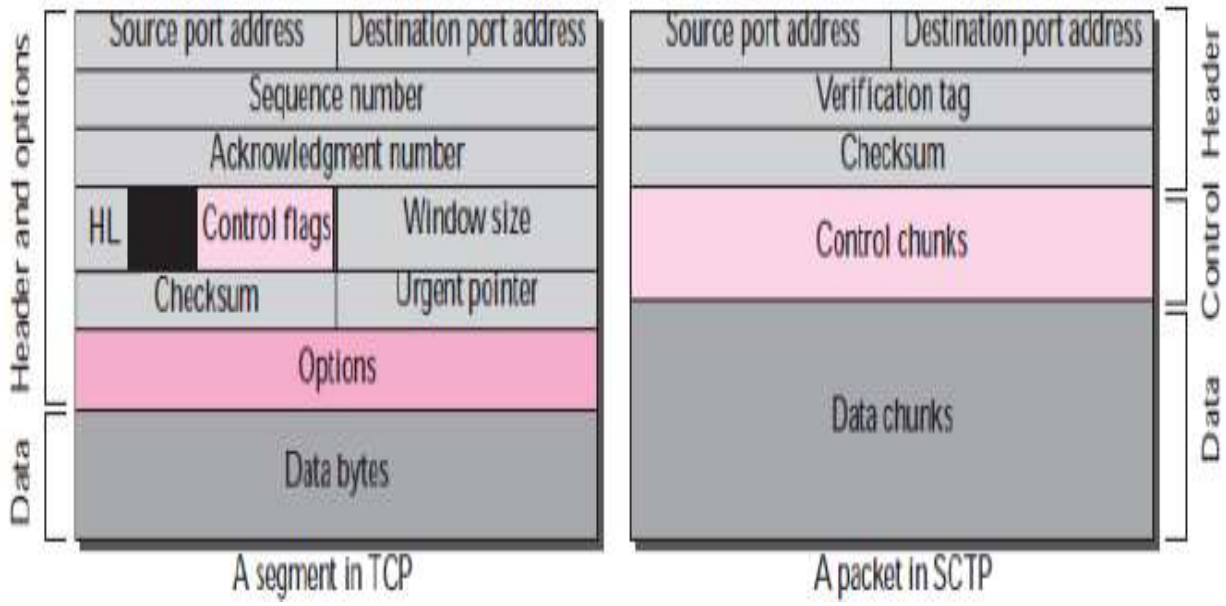
• In TCP, a segment carries data and control information. Data are carried as a collection of bytes; control information is defined by six control flags in the header.

• 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.



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.

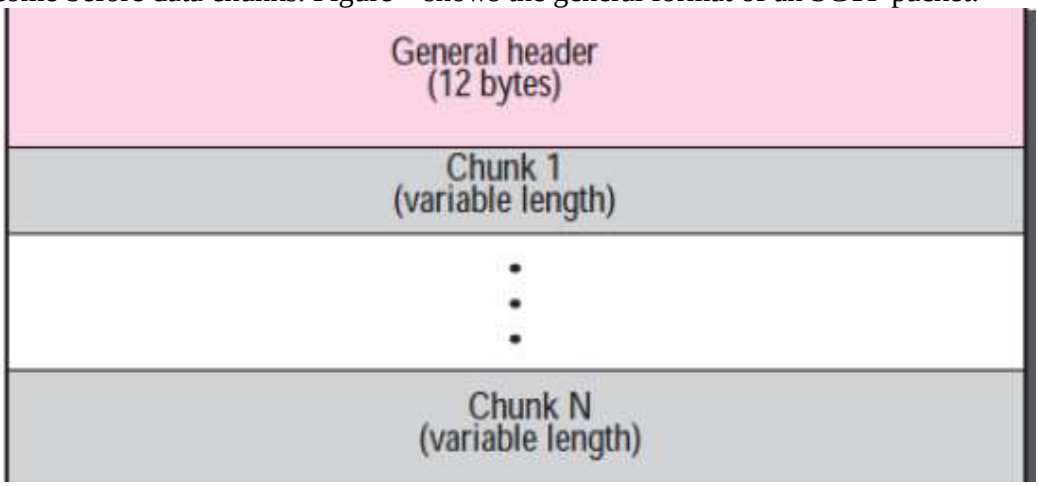
Flow Control –

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

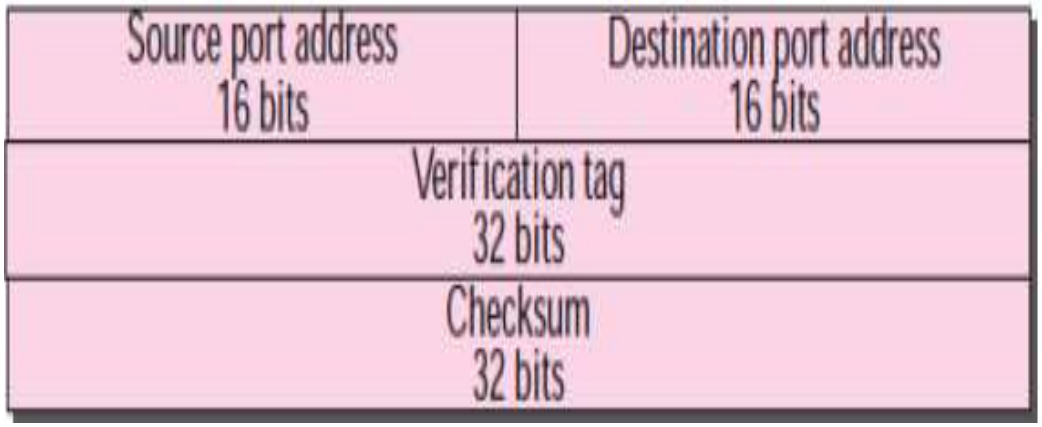
Error Control –
• Like TCP, SCTP implements error control to provide reliability. TSN numbers and acknowledgment numbers are used for error control.

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.

UNIT :- IV

Host Configuration: DHCP

INTRODUCTION

Each computer that uses the TCP/IP protocol suite needs to know its IP address.
If the computer uses classless addressing or is a member of a subnet, it also needs to know its subnet mask.
Most computers today need two other pieces of information: the address of a default router to be able to communicate with other networks and the address of a name server to be able to use names instead of addresses.
. In other words, four pieces of information are normally needed:

1. The IP address of the computer
2. The subnet mask of the computer
3. The IP address of a router
4. The IP address of a name server

These four pieces of information can be stored in a configuration file and accessed by the computer during the bootstrap process. But what about a diskless workstation or a computer with a disk that is booted for the first time?
In the case of a diskless computer, the operating system and the networking software could be stored in read-only memory (ROM).
However, the above information is not known to the manufacturer and thus cannot be stored in ROM.
The information is dependent on the individual configuration of the machine and defines the network to which the machine is connected.

Previous Protocols

Before DHCP became the formal protocol for host configuration, some other protocols were used for this propose. We briefly describe them here.

RARP

At the beginning of the Internet era, a protocol called Reverse Address Resolution Protocol (RARP) was designed to provide the IP address for a booted computer.
RARP was actually a version of ARP.
ARP maps an IP address to a physical address: RARP maps a physical address to an IP address. However, RARP is deprecated today for two reasons.
First, RARP used the broadcast service of the data link layer, which means that a RARP server must be present in each network. Second, RARP can provide only the IP address of the computer, but a computer today needs all our pieces of information mentioned above.

BOOTP

The **Bootstrap Protocol (BOOTP)** is the prerunner of DHCP. It is a client/server protocol designed to overcome the two deficiencies of the RARP protocol. First, since it is a client/server program, the BOOTP server can be anywhere in the Internet. Second, it can provide all pieces of information we mentioned above, including the IP address.
To provide the four pieces of information described above, it removes all restriction about the RARP protocol.
BOOTP, however, is a *static configuration protocol.*
When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address.
This implies that the binding between the physical address and the IP address of the client already exists.
The binding is predetermined. here are some situations in which we need a *dynamic configuration protocol*.
For example, when a host moves from one physical network to another, its physical address changes.
As another example, there are occasions when a host wants a temporary IP address to be used for a period of time.
BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator.
As we will see shortly, DHCP has been devised to handle these shortcomings.

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 is a successor to BOOTP and is backward compatible with it.

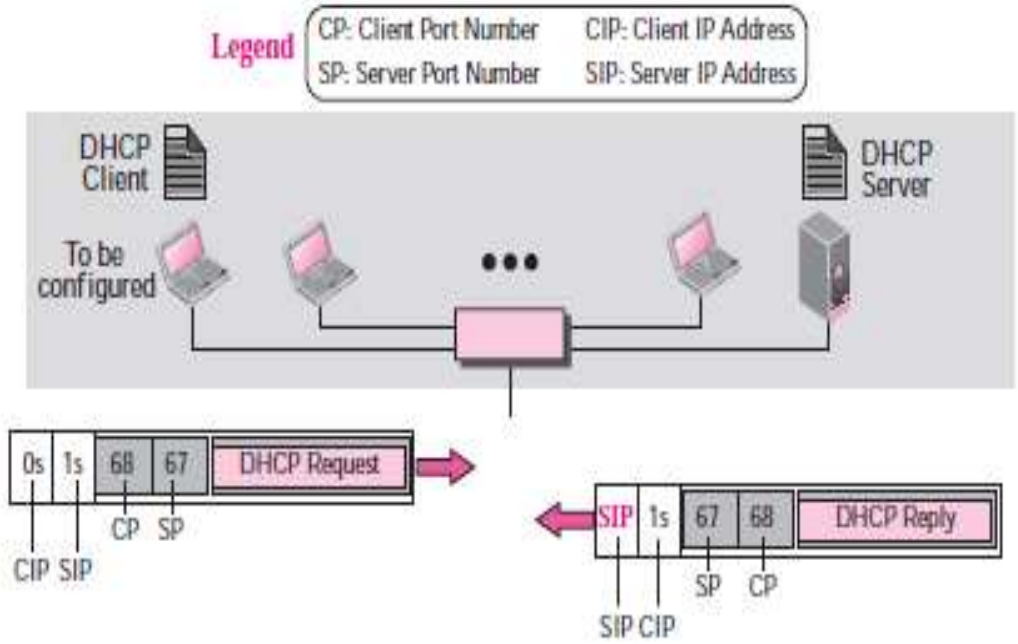
Although BOOTP is considered deprecated, there may be some systems that may still use BOOTP for host configuration.
The part of the discussion in this chapter that does not deal with the dynamic aspect of DHCP can also be applied to BOOTP.

DHCP OPERATION

The DHCP client and server can either be on the same network or on different networks. Let us discuss each situation separately.

Same Network

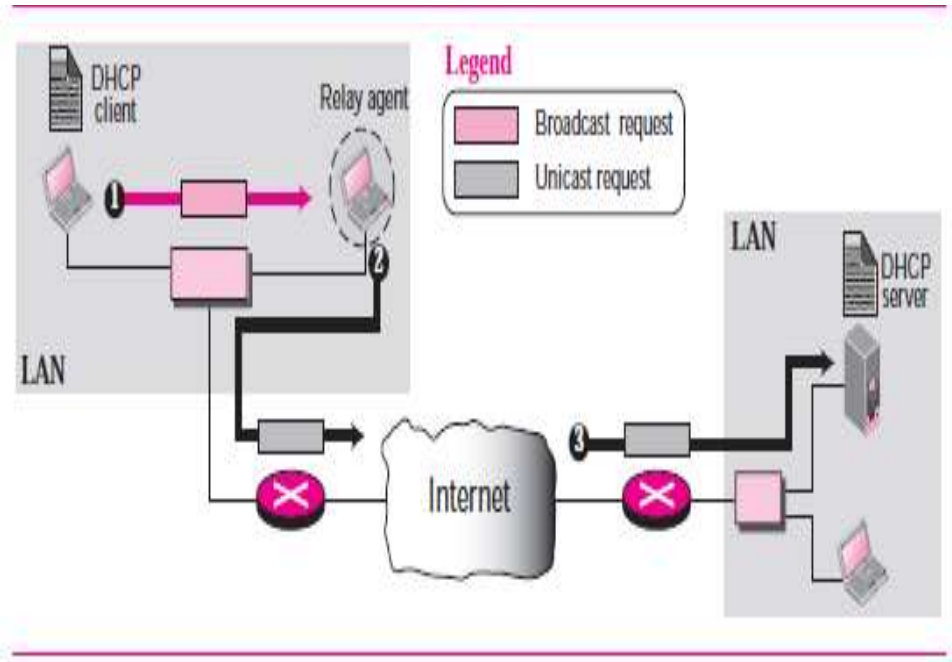
Although the practice is not very common, the administrator may put the client and the server on the same network as shown in Figure



In this case, the operation can be described 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 (this number will be explained later).
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

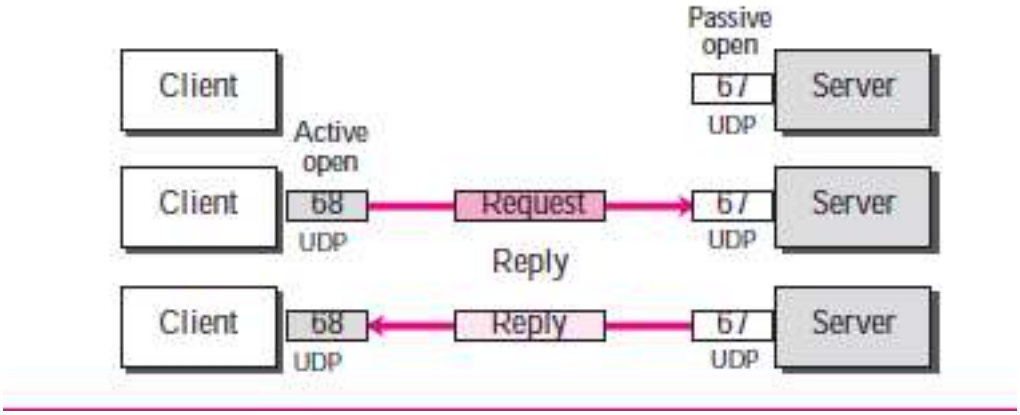
As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks.



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.
Recall that an IP address of all 1s is a limited broadcast address.
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

UDP Ports

The server uses the well-known port 67, which is normal.
The client uses the well-known port 68, which is unusual.
The reason for choosing the well-known port 68 instead of an ephemeral port is to prevent a problem when the reply, from the server to the client, is broadcast.
To understand the problem, let us look at a situation where an ephemeral port is used.
Suppose host A on a network is using a DHCP client on ephemeral port 2017 (randomly chosen).
Host B, on the same network, is using a DAYTIME client on ephemeral port 2017 (accidentally the same).
Now the DHCP server sends a broadcast reply message with the destination port number 2017 and broadcast IP address $FFFFFFFF_{16}$.
Every host needs to open a packet carrying this destination IP address.
Host A finds a message from an application program on ephemeral port 2017. A correct message is delivered to the DHCP client.
An incorrect message is delivered to the DAYTIME client



The use of a well-known port (less than 1024) prevents the use of the same two destination port numbers.
Host B cannot select 68 as the ephemeral port because ephemeral port numbers are greater than 1023.
The curious reader may ask what happens if host B is also running the DHCP client. In this case, the socket address is the same and both clients will receive the message.
In this situation, a third identification number differentiates the clients.
DHCP uses another number, called the transaction ID, which is randomly chosen for each connection involving DHCP.
It is highly improbable that two hosts will choose the same ID at the same time.

Using TFTP

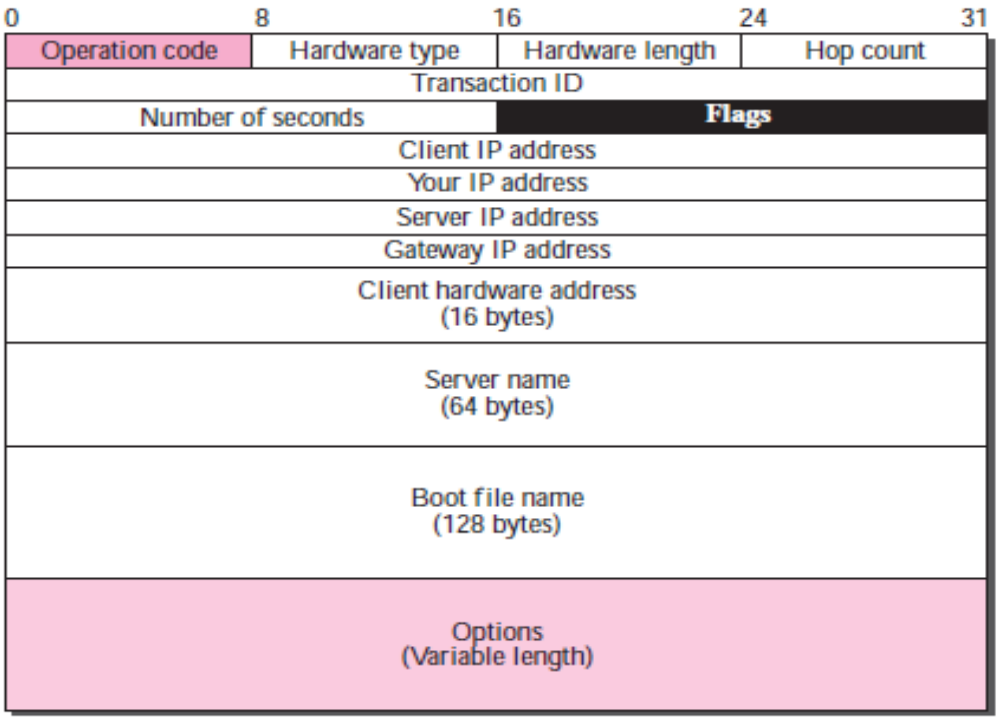
The server does not send all of the information that a client may need for booting.
In the reply message, the server defines the pathname of a file in which the client can findcomplete booting information.
The client can then use a TFTP message which is encapsulated in a UDP user datagram, to obtain the rest of the needed
information.

Error Control

What if a request is lost or damaged? What if the response is damaged?
There is a need for error control when using DHCP. DHCP uses UDP, which does not provide error control. Therefore, DHCP must provide error control.
Error control is accomplished through two strategies:
1. DHCP requires that UDP uses the checksum. Remember that the use of the checksum in UDP is optional.
2. The DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request. However, to prevent a traffic jam when several hosts need to retransmit a request (for example, after a power failure), DHCP forces the client
to use a random number to set its timers.

Packet Format

DHCP packet.



Operation code. This 8-bit field defines the type of DHCP packet: request (1) or reply (2).

Hardware type. This is an 8-bit field defining the type of physical network. Each type of network has been assigned an integer. For example, for Ethernet the value is 1.

- **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.
- **Hop count.** This is an 8-bit field defining the maximum number of hops the packet can travel.
- **Transaction ID.** This is a 4-byte field carrying an integer. The transaction identification is set by the client and is used to match a reply with the request. The server returns the same value in its reply.
- **Number of seconds.** This is a 16-bit field that indicates the number of seconds elapsed since the time the client started to boot.
- **Flag.** This is a 16-bit field in which only the leftmost bit is used and the rest of the bits should be set to 0s. A leftmost bit specifies a forced broadcast reply (instead of unicast) from the server. If the reply were to be unicast to the client, the destination

IP address of the IP packet is the address assigned to the client. Since the client does not know its IP address, it may discard the packet. However, if the IP datagram is broadcast, every host will receive and process the broadcast message.

Client IP address. This is a 4-byte field that contains the client IP address. If the client does not have this information, this field has a value of 0.

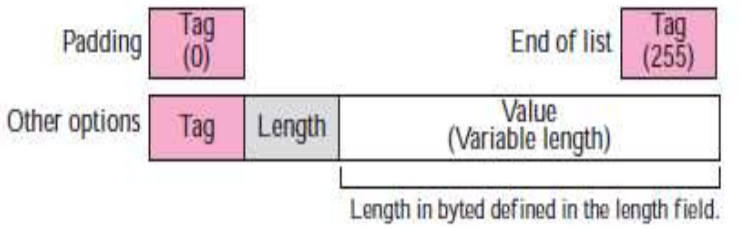
- **Your IP address.** This is a 4-byte field that contains the client IP address. It is filled by the server (in the reply message) at the request of the client.
- **Server IP address.** This is a 4-byte field containing the server IP address. It is filled by the server in a reply message.
- **Gateway IP address.** This is a 4-byte field containing the IP address of a router. It is filled by the server in a reply message.

❑ **Client hardware address.** This is the physical address of the client. Although the server can retrieve this address from the frame sent by the client, it is more efficient if the address is supplied explicitly by the client in the request message.

❑ **Server name.** This is a 64-byte field that is optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the domain name of the server. If the server does not want to fill this field with data, the server must fill it
with all 0s.

❑ **Boot filename.** This is a 128-byte field that can be optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the full pathname of the boot file. The client can use this path to retrieve other booting information. If the server does not want to fill this field with data, the server must fill it with all 0s.

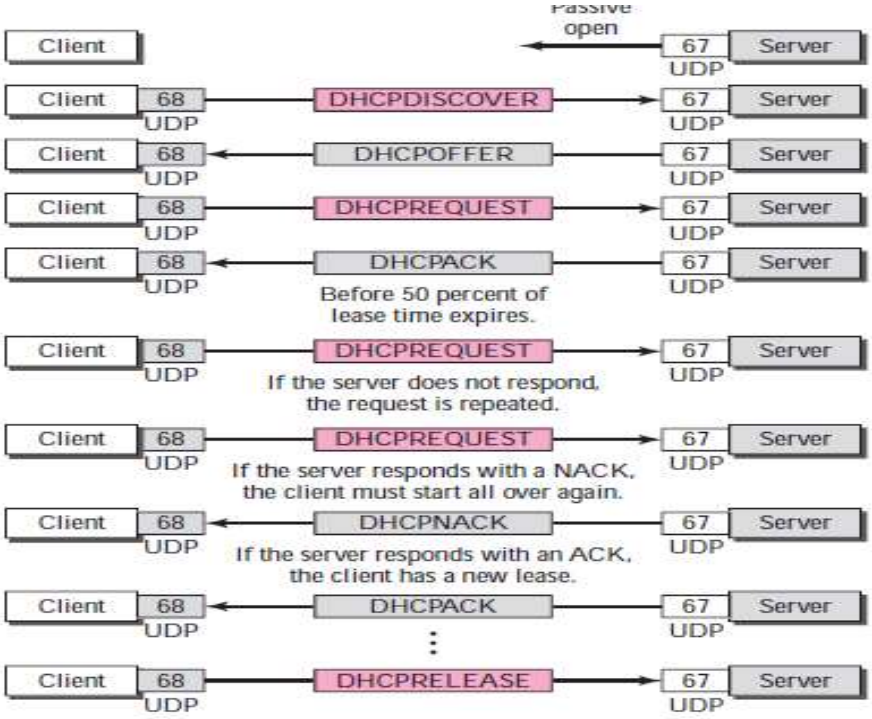
❑ **Options.** This is a 64-byte field with a dual purpose. It can carry either additional information (such as the network mask or default router address) or some specific vendor information. The field is used only in a reply message. The server uses a number,
called a **magic cookie,** in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields:
a 1-byte tag field, a 1-byte length field, and a variable-length value field. The length field defines the length of the value field, not the whole option



Tag	*Length*	*Value*	*Description*
0			Padding
1	4	Subnet mask	Subnet mask
2	4	Time of the day	Time offset
3	Variable	IP addresses	Default router
4	Variable	IP addresses	Time server
5	Variable	IP addresses	IEN 16 server
6	Variable	IP addresses	DNS server
7	Variable	IP addresses	Log server
8	Variable	IP addresses	Quote server
9	Variable	IP addresses	Print server
10	Variable	IP addresses	Impress
11	Variable	IP addresses	RLP server
12	Variable	DNS name	Host name
13	2	Integer	Boot file size
53	1	Discussed later	Used for dynamic configuration
128–254	Variable	Specific information	Vendor specific
255			End of list

Exchanging Messages

Figure shows the exchange of messages related to the transition diagram.


Passive open
Client
67
UDP
Server
Client
68
UDP
DHCPDISCOVER
67
UDP
Server
Client
68
UDP
DHCPOFFER
67
UDP
Server
Client
68
UDP
DHCPREQUEST
67
UDP
Server
Client
68
UDP
DHCPACK
67
UDP
Server
Before 50 percent of lease time expires.
Client
68
UDP
DHCPREQUEST
67
UDP
Server
If the server does not respond, the request is repeated.
Client
68
UDP
DHCPREQUEST
67
UDP
Server
If the server responds with a NACK, the client must start all over again.
Client
68
UDP
DHCPNACK
67
UDP
Server
If the server responds with an ACK, the client has a new lease.
Client
68
UDP
DHCPACK
67
UDP
Server
Client
68
UDP
DHCPRELEASE
67
UDP
Server

Domain Name System (DNS)

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).**
In this chapter, we first discuss the concepts and ideas behind the DNS. We then describe the DNS protocol itself.

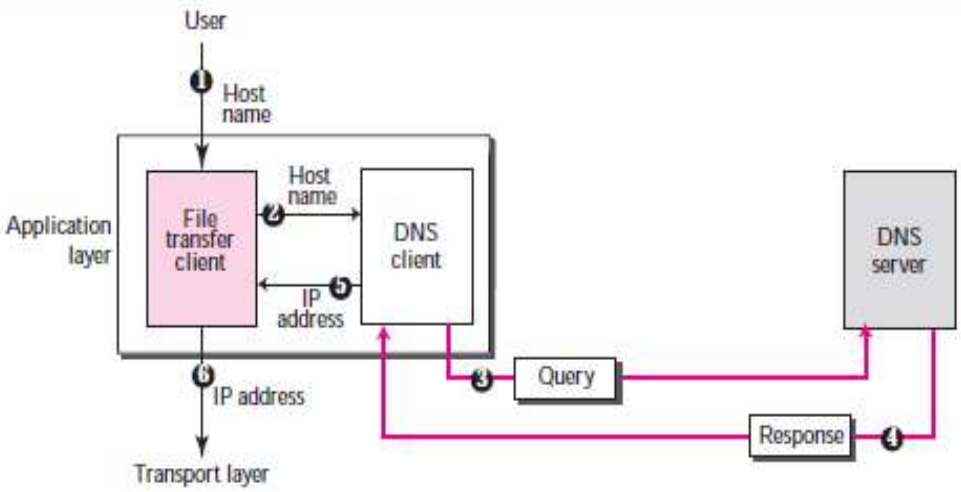


Diagram :- Purpose of DNS

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 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.

Note that the purpose of accessing the Internet is to make a connection between the file transfer client and server, but before this can happen, another connection needs to be made between the DNS client and DNS server.
In other words, we need two connections; the first is for mapping the name to an IP address; the second is for transferring files

NAME SPACE

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

Flat Name Space

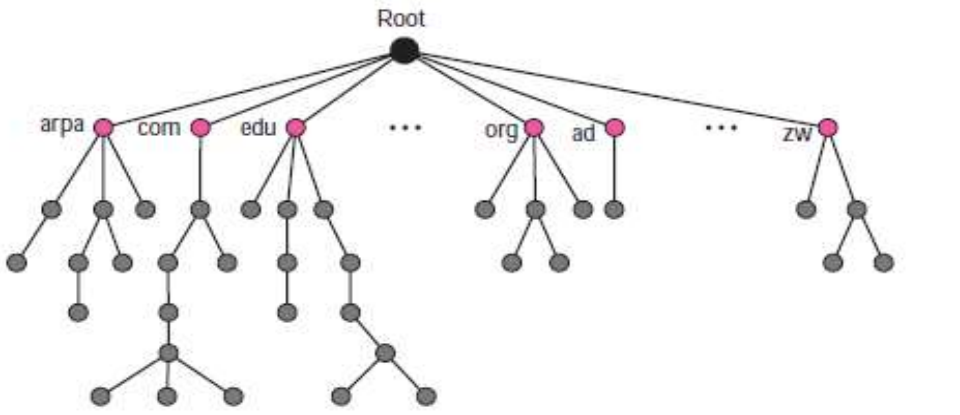
In a **flat name space,** a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is
that it cannot be used in a large system such as 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 ca 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 an the name of the organization. The responsibility of the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the same, the whole address is different. For example, assume two colleges and a company call one of their computers *challenger*. The first college is given a name by the central authority such as *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,* an *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.



Label

Each node in the tree has a **label,** which is a string with a maximum of 63 characters.

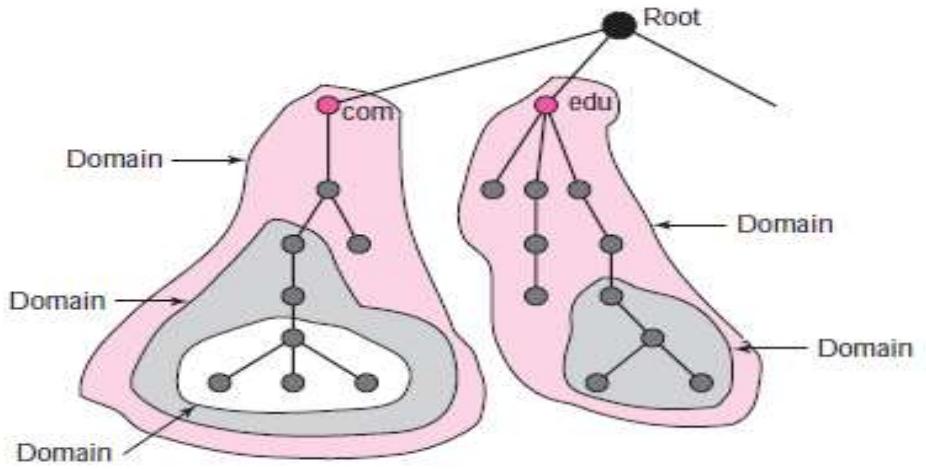
The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name

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

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 19.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes
called).

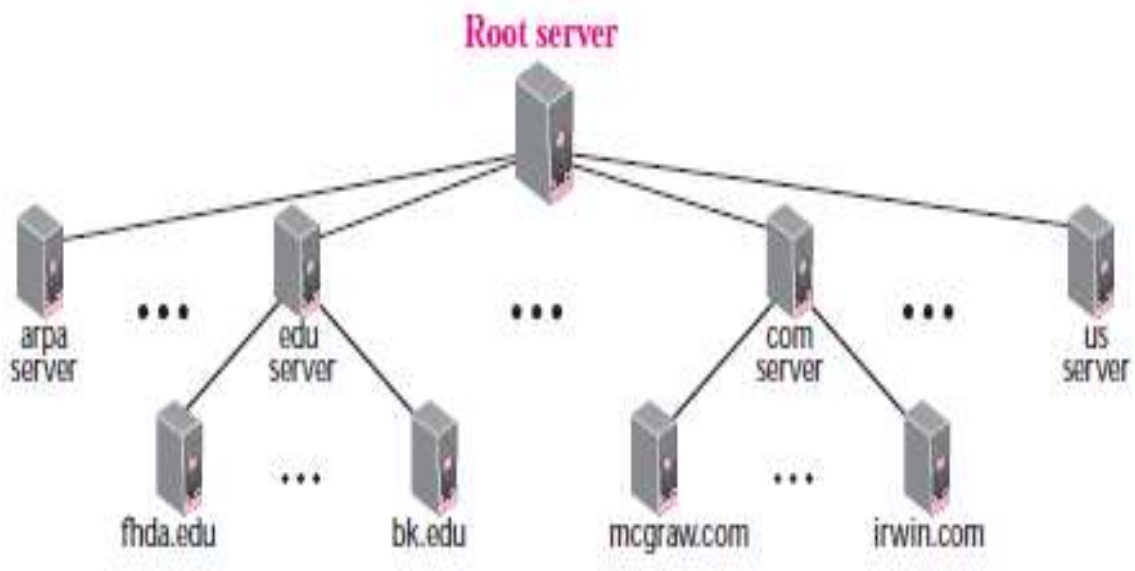


Distribution of Name Space

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

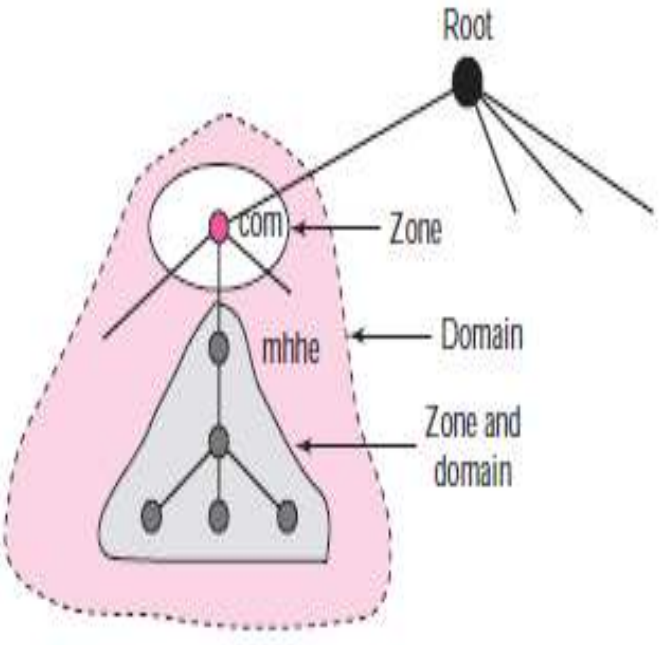
The solution to these problems is to distribute the information among many computers called **DNS servers.** One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as man domains (subtrees) as there are first-level 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



Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a **zone.** We can define a zone as a contiguous part of the entire tree. If a server
accepts responsibility for a domain and does not divide the domain into smaller domains, the "domain" and the "zone" refer to the same thing. The server makes a database called a *zone file* and keeps all the information for every node under that domain.

However, if a server divides its domain into subdomains and delegates part of its authority to other servers, "domain" and "zone" refer to different things. The information about the nodes in the subdomains is stored in the servers at the lower levels, with
the original server keeping some sort of reference to these lower-level servers.



A server can also divide part of its domain and delegate responsibility but still keep part of the domain for itself. In this case, its zone is made of detailed information for the part of the domain that is not delegated and references to those parts that are delegated.

Root Server

A **root server** is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space. The root servers are distributed all around the world.

Primary and Secondary Servers

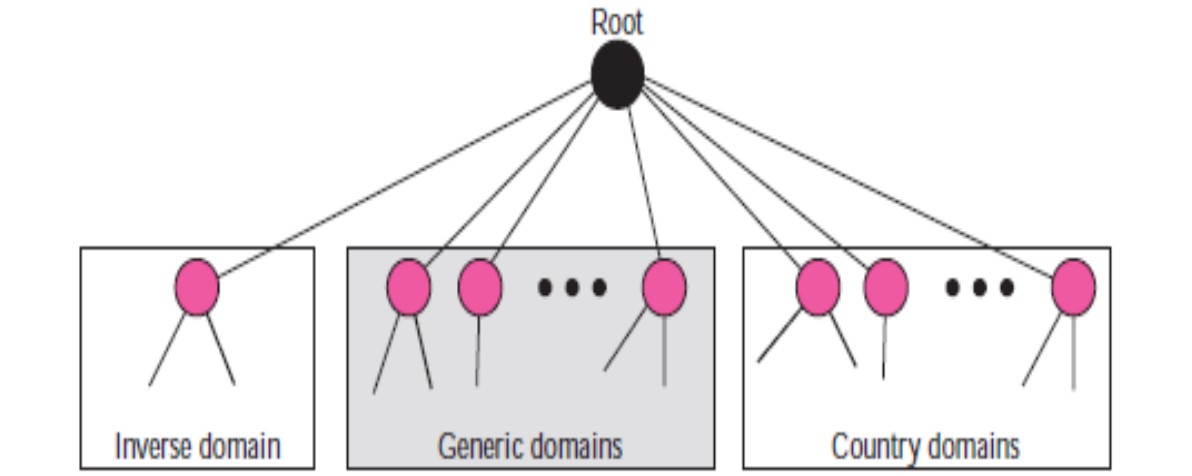
DNS defines two types of servers: primary and secondary. A **primary server** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk. A **secondary server** is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is
required, it must be done by the primary server, which sends the updated version to the secondary. The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients. Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful about which zone we refer to.

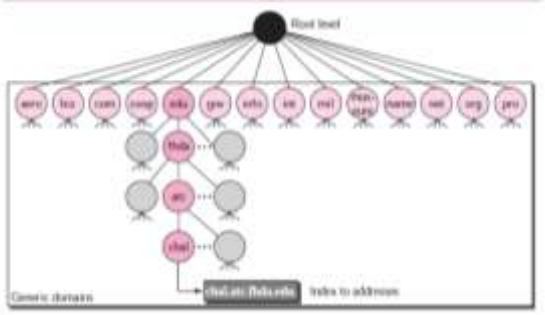
DNS IN THE INTERNET

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

Generic Domains

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

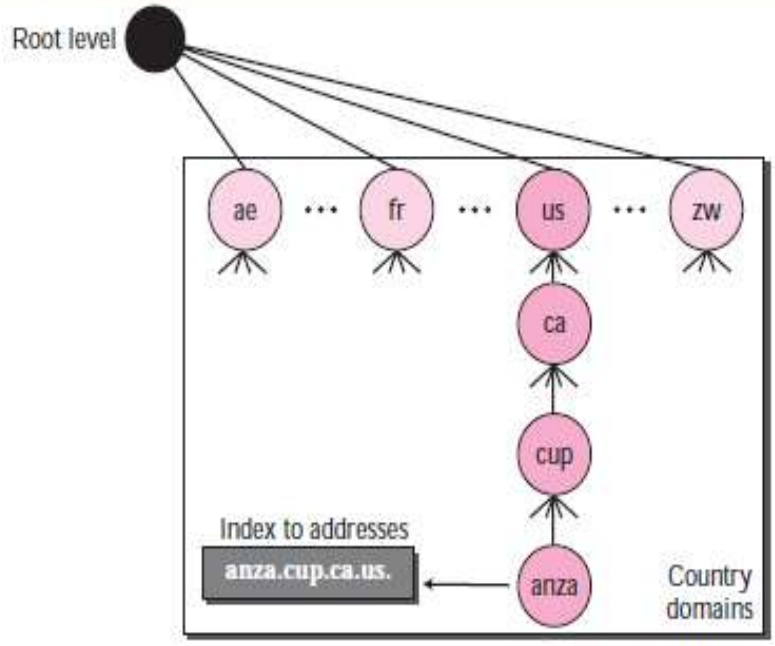




Country Domains

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

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



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.

Remote Login: TELNET and SSH

The main task of the Internet and its TCP/IP protocol suite is to provide services for users. Although there are some specific client/server programs that we discuss in future chapters, it would be impossible to write a specific client-server program for each demand. The better solution is a general-purpose client-server program that lets a user access any application program on a remote computer; in other words, allow the user to log on to a remote computer. After logging on, a user can use the services
available on the remote computer and transfer the results back to the local computer. In this chapter, we discuss two of these application programs: TELNET and SSH.

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.

Concepts

TELNET is related to several concepts that we briefly describe here.

Time-Sharing Environment

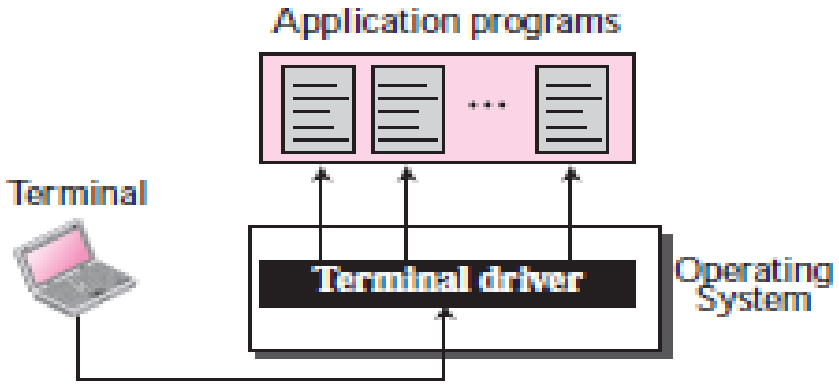
TELNET was designed at a time when most operating systems, such as UNIX, were operating in a **time-sharing** environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs
through a terminal, which is usually a combination of keyboard, 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 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

Local login



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. Whereas these special situations do not create any problem in local login because the terminal emulator and the terminal driver know the exact meaning of each character or combination of characters, they may create problems in remote login. Which process should interpret special characters? The client or the server? We will clarify this situation later in the chapter.

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

Figure :- Remote Login

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

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-duple with the client sending one line after another, without the need for an intervening GA (go ahead) character from the server.

SECURE SHELL (SSH)

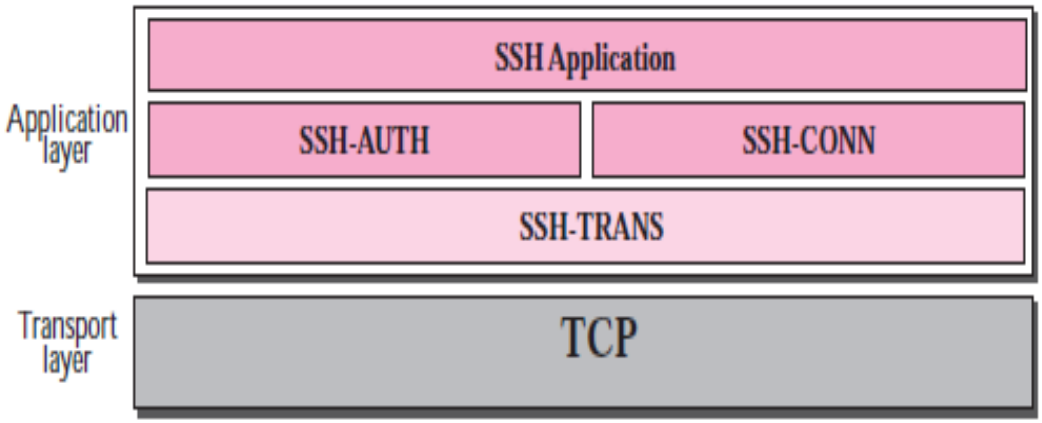
Another popular remote login application program is **Secure Shell (SSH)**. SSH, like TELNET, uses TCP as the underlying transport protocol, but SSH is more secure and provides more services than TELNET.

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. In this section, we discuss only SSH-2.

Components

SSH is a proposed application-layer protocol with four components,



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.
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)

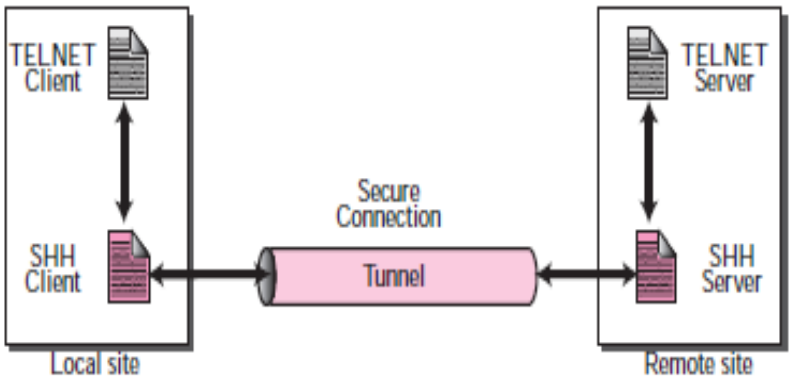
After the secured channel is established and both server and client are authenticated for each other, SHH can call a piece of software that implements the third protocol, SSHCONN. One of the services provided by the SSH-CONN protocol is to do multiplexing. SSH-CONN takes the secure channel established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

SSH Applications

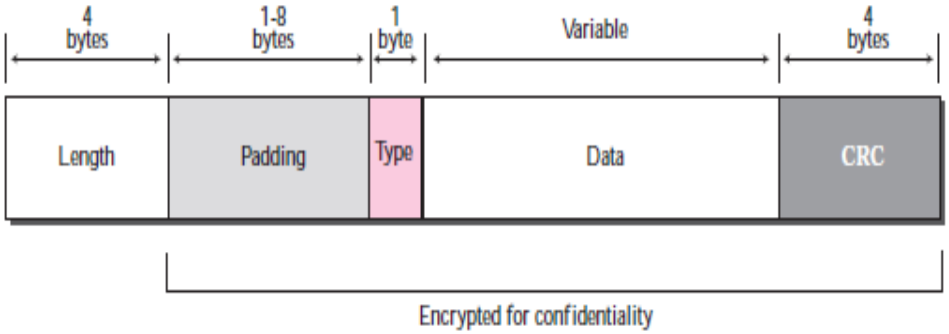
After the connection phase is completed, SSH allows several application programs to use the connection. Each application can create a logical channel as described above and then benefit from the secured connection. In other words, remote login is one of the services that can use the SSH-CONN protocols;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

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.



Format of the SSH Packets



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

File Transfer: FTP and TFTP

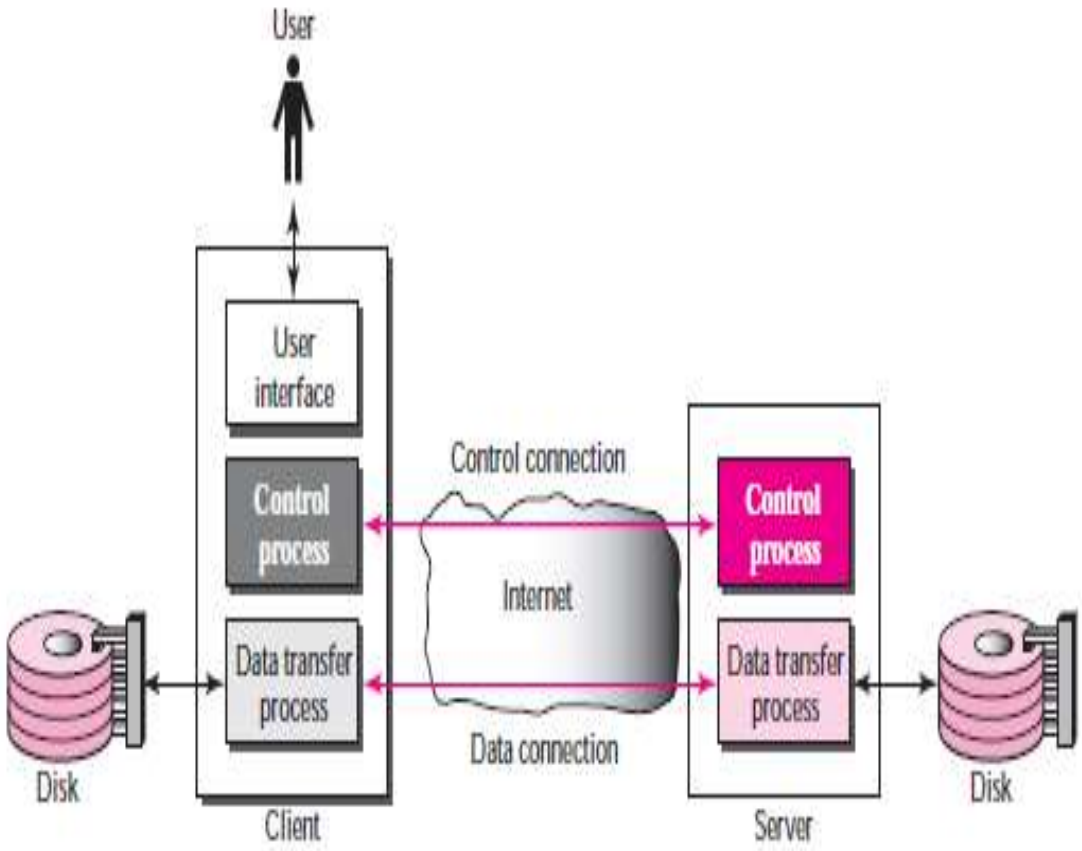
Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to file transfer. In this chapter, we discuss two protocols involved in transferring files: File Transfer Protocol (FTP) and Trivial File Transfer Protocol (TFTP).

FTP

File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another. Although transferring files from one system t another seems simple and straightforward, some problems must be dealt with first. Fo example, two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. All of these problems have been solved by FTP in a very simple and elegan approach. FTP differs from other client-server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. 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. 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 described so far. 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

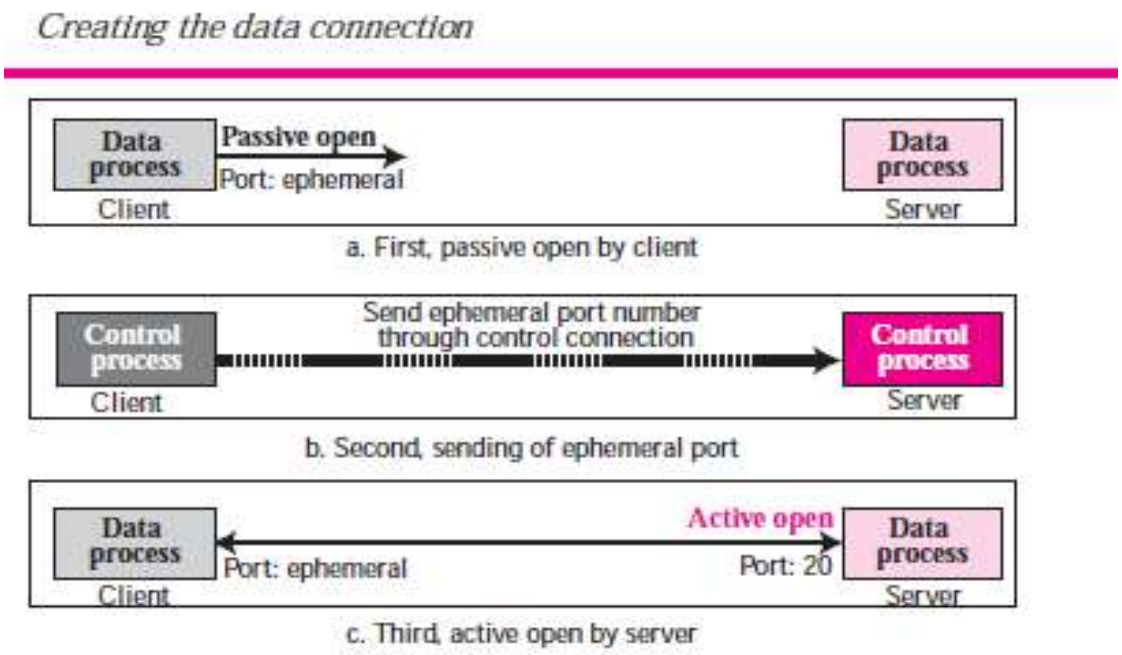
Data Connection

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.

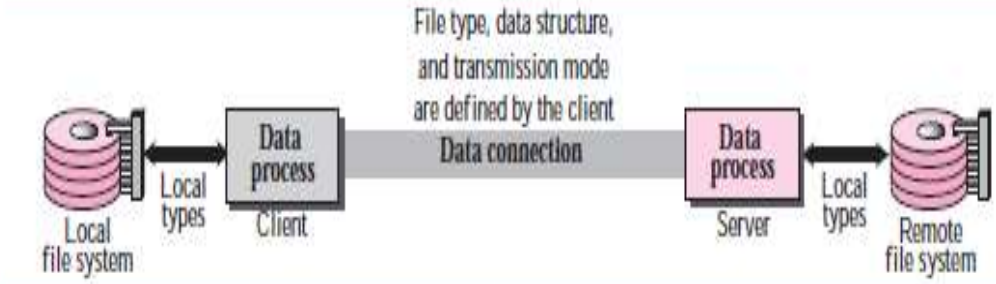
2. The client sends this port number to the server using the PORT command (we will discuss this command shortly).

3. The server receives the port number and issues an active open using the wellknown port 20 and the received ephemeral port number



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



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 printe 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.

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.

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

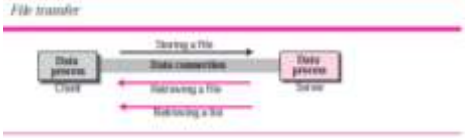
- **Stream mode.** This is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data is simply a stream of bytes (file structure), no end-offile is needed. End-of-file in this case is the closing 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.
- **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.
- **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.

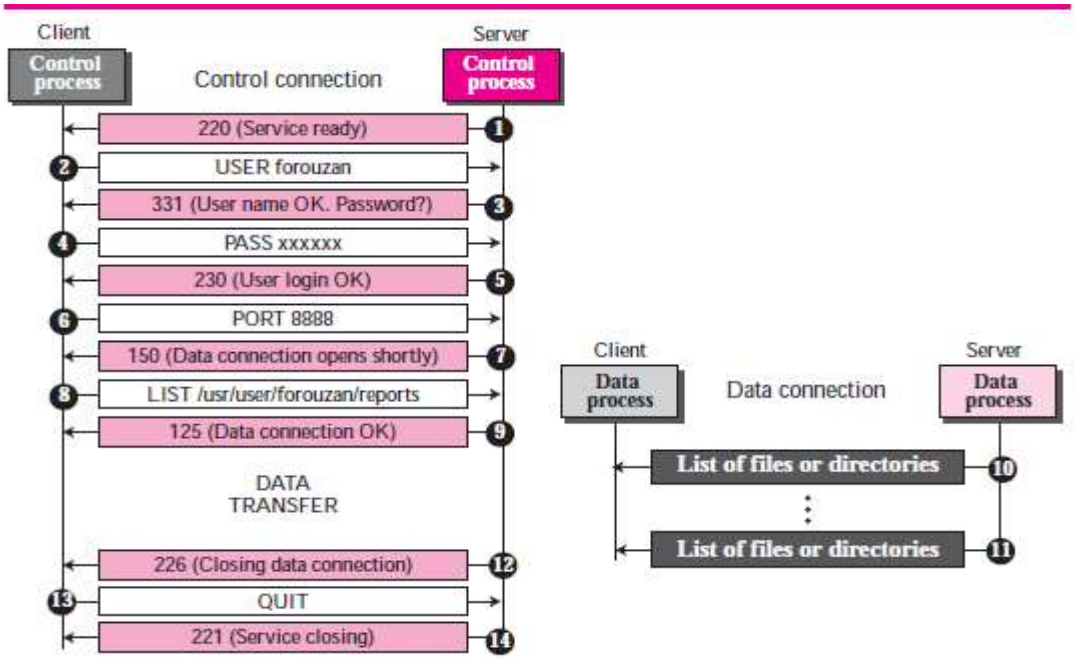
In a text file, this is

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

- 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





1. After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.
2. The client sends the USER command.
3. The server responds with 331 (user name is OK, password is required).
4. The client sends the PASS command.
5. The server responds with 230 (user login is OK).
6. The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.
7. The server does not open the connection at this time, but it prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends response 150 (data connection will open shortly).
8. The client sends the LIST message.
9. Now the server responds with 125 and opens the data connection.
10. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
11. The client now has two choices. It can use the QUIT command to request the closing of the control connection or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
12. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control 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.

Messages

There are five types of TFTP messages, RRQ, WRQ, DATA, ACK, and ERROR,

RRQ

The read request (RRQ) message is used by the client to establish a connection for reading data from the server. 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.

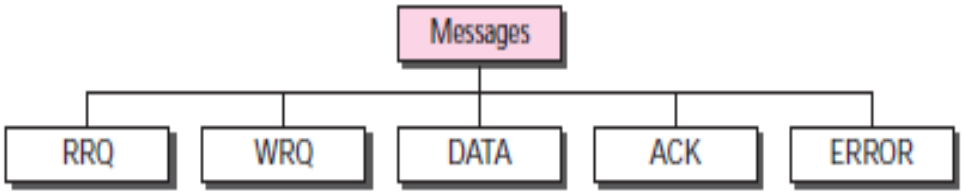
❑ **File name.** The next field is a variable-size string (encoded in ASCII) that defines the name of the file. Since the file name varies in length, termination is signaled by a 1-byte field of 0s.

❑ **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

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



RRQ

The read request (RRQ) message is used by the client to establish a connection for reading data from the server

RRQ format

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

WRQ format

DATA

The data (DATA) message is used by the client or the server to send blocks of data. Its format is shown in Figure 21.13. 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

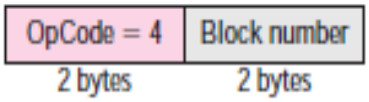
. ❑ **Block number.** This is a 2-byte field containing the block number. The sender of the data (client or server) uses this field for sequencing. All blocks are numbered sequentially starting with 1. The block number is necessary for acknowledgment as we will see shortly.

❑ **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 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.

ACK format



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. An example of an ACK message is given in a later section.

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 discussed later in this chapter 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 21.8 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.

UNIT :- V

World Wide Web and HTTP

The **World Wide Web (WWW)** is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research. In this chapter we first discuss issues related to the Web. We then discuss a protocol, HTTP, that is used to retrieve information from the Web.

ARCHITECTURE

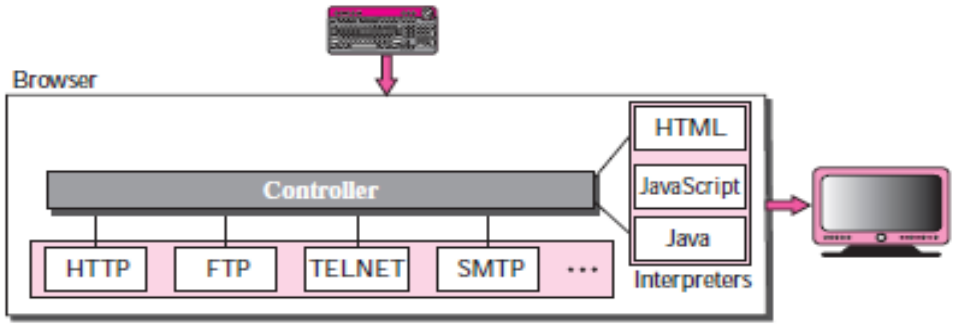
The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites.* Each site holds one or more documents, referred to as Web pages. Each **Web page**, however, can contain some links to other Web pages in the same or other sites. In other words, a Web page can be simple or composite. 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.

Hypertext and Hypermedia

The three previous examples show the idea of **hypertext** and **hypermedia**. Hypertext means creating documents that refer to other documents. In a hypertext document, a part of text can be defined as a link to another document. When a hypertext is viewed with a browser, the link can be clicked to retrieve the other document. Hypermedia is a term applied to document that contains links to other textual document or documents containing graphics, video, or audio.

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



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 (as discussed later in the chapter). The interpreter can be HTML, Java, or JavaScript, depending on the type of document. We discuss the use of these interpreters based on the document type later in the chapter. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

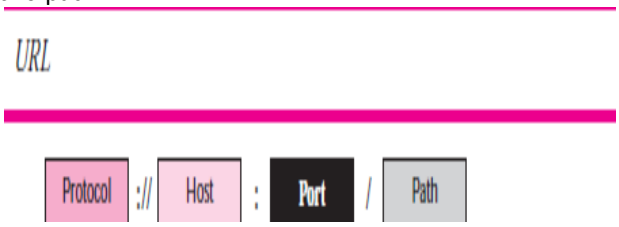
Web Server

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



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. The category is based on the time the contents of the document are determined.

Static Documents

Static documents are fixed-content documents that are created and stored in a server. The client can get 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

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. CGI is not a new language; instead, it allows programmers to use any of several languages such as C, C ++, Bourne Shell, Korn Shell, C Shell, Tcl, or Perl. The only thing that CGI defines is a set of rules and terms that the programmer must follow.
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. A CGI program in its simplest form is code written in one of the languages supporting CGI. Any programmer who can

encode a sequence of thoughts in a program and knows the syntax of one of the abovementioned languages can write a simple CGI program.

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called **active documents.** For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place. When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client (browser) site.

Java Applets

One way to create an active document is to use **Java applets. Java** is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document (an applet) and a browser to run it. It can also be a stand-alone program that doesn't use a browser. An **applet** is a program written in Java on the server. It is compiled and ready to be run. The document is in bytecode (binary) format. The client process (browser) creates
an instance of this applet and runs it. A Java applet can be run by the browser in two ways. In the first method, the browser can directly request the Java applet program in the URL and receive the applet in binary form. In the second method, the browser can
retrieve and run an HTML file that has embedded the address of the applet as a tag.

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 22.9 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

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

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 . 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, **Active documents are sometimes referred to as client-site dynamic documents. HTTP uses the services of TCP on well-known port 80.**

HTTP Transaction

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

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. 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,

Cookies

The World Wide Web was originally designed as a stateless entity. A client sends a request; a server responds. Their relationship is over. The original design of WWW, retrieving publicly available documents, exactly fits this purpose. Today the Web has
other functions; some are listed below:

- ❑ Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
- ❑ Some websites need to allow access to registered clients only.
- ❑ Some websites are used as portals: The user selects the Web pages he wants to see.
- ❑ Some websites are just advertising. For these purposes, the cookie mechanism was devised.

Creating and Storing Cookies

The creation and storing of cookies depend on the implementation; however, the principle is the same.

1. When a server receives a request from a client, it stores information about the client in a file or a string. The information may include the domain name of the client, the contents of the cookie (information the server has gathered about the client such as
name, registration number, and so on), a timestamp, and other information depending on the implementation.

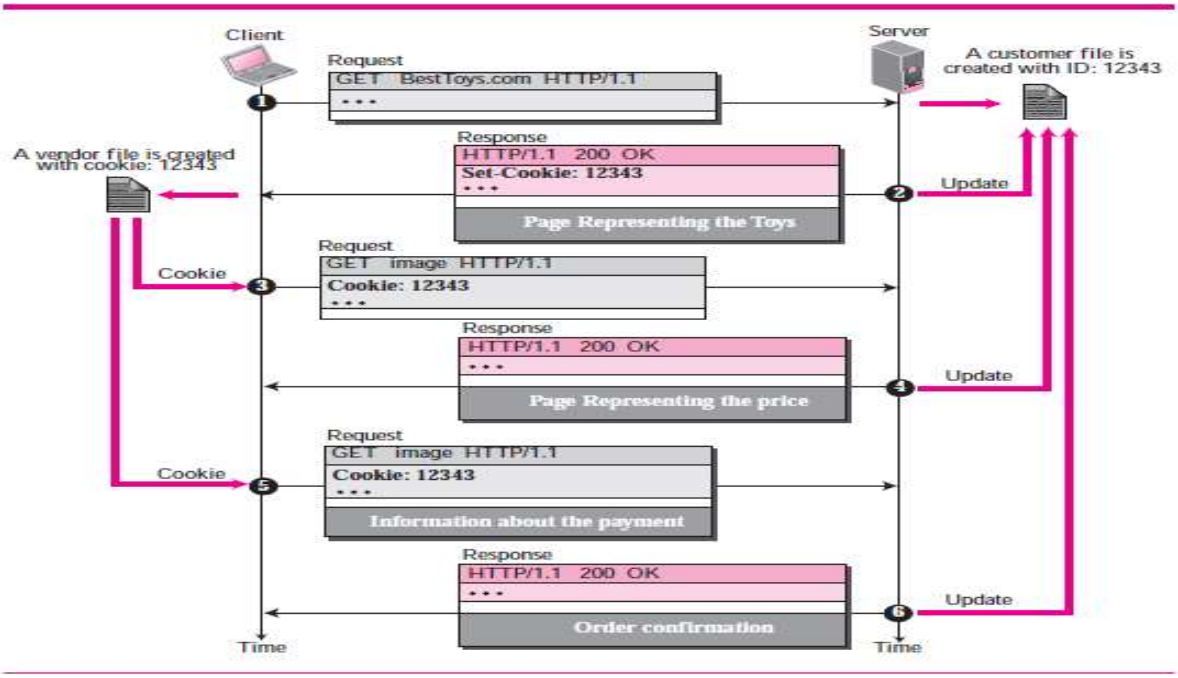
2. The server includes the cookie in the response that it sends to the client.

3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the domain server name.

Using Cookies

When a client sends a request to a server, the browser looks in the cookie directory to see if it can find a cookie sent by that server. If found, the cookie is included in the request. When the server receives the request, it knows that this is an old client, not a new one. Note that the contents of the cookie are never read by the browser or disclosed to the user. It is a cookie *made* by the server and *eaten* by the server. Now let us see how a cookie is used for the four previously mentioned purposes:

- ❑ An *electronic store* (e-commerce) can use a cookie for its client shoppers. When a client selects an item and inserts it into a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser. If the client
selects a second item, the cookie is updated with the new selection information. And so on. When the client finishes shopping and wants to check out, the last cookie is retrieved and the total charge is calculated.
- ❑ The site that restricts access to *registered clients* only sends a cookie to the client when the client registers for the first time. For any repeated access, only those clients that send the appropriate cookie are allowed.
- ❑ A Web *portal* uses the cookie in a similar way. When a user selects her favorite pages, a cookie is made and sent. If the site is accessed again, the cookie is sent to the server to show what the client is looking for.
- ❑ A cookie is also used by *advertising* agencies. An advertising agency can place banner ads on some main website that is often visited by users. The advertising agency supplies only a URL that gives the banner address instead of the banner
itself. When a user visits the main website and clicks the icon of an advertised corporation, a request is sent to the advertising agency. The advertising agency sends the banner, a GIF file for example, but it also includes a cookie with the ID of the
user. Any future use of the banners adds to the database that profiles the Web behavior of the user. The advertising agency has compiled the interests of the user and can sell this information to other parties. This use of cookies has made them
very controversial. Hopefully, some new regulations will be devised to preserve the privacy of users.


Client
Server
A customer file is created with ID: 12343
Request
GET BestToys.com HTTP/1.1
• • •
A vendor file is created with cookie: 12343
Response
HTTP/1.1 200 OK
Set-Cookie: 12343
• • •
Page Representing the Toys
Update
Cookie
Request
GET image HTTP/1.1
Cookie: 12343
• • •
Response
HTTP/1.1 200 OK
• • •
Page Representing the price
Update
Cookie
Request
GET image HTTP/1.1
Cookie: 12343
• • •
Information about the payment
Response
HTTP/1.1 200 OK
• • •
Order confirmation
Update
Time
Time

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. Its architecture consists of several components that we will discuss in this chapter.

At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients. In this chapter, we first study the general architecture of an e-mail

system including the three main components: user agent, message transfer agent, and message access agent. We then describe the protocols that implement these components

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

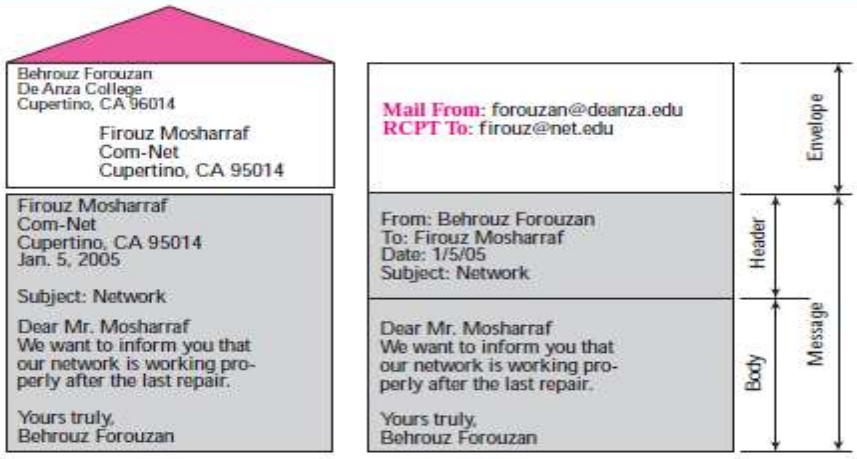
A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers

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 onecharacter 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. 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.

Sending Mail

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



Envelope

The **envelope** usually contains the sender address, the receiver address, and other information.

Message

The message contains the **header** and the **body.** The header of the message defines the sender, the receiver, the subject of the message, and some other information. The body of the message contains the actual information to be read by the recipient.

Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which

each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

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

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

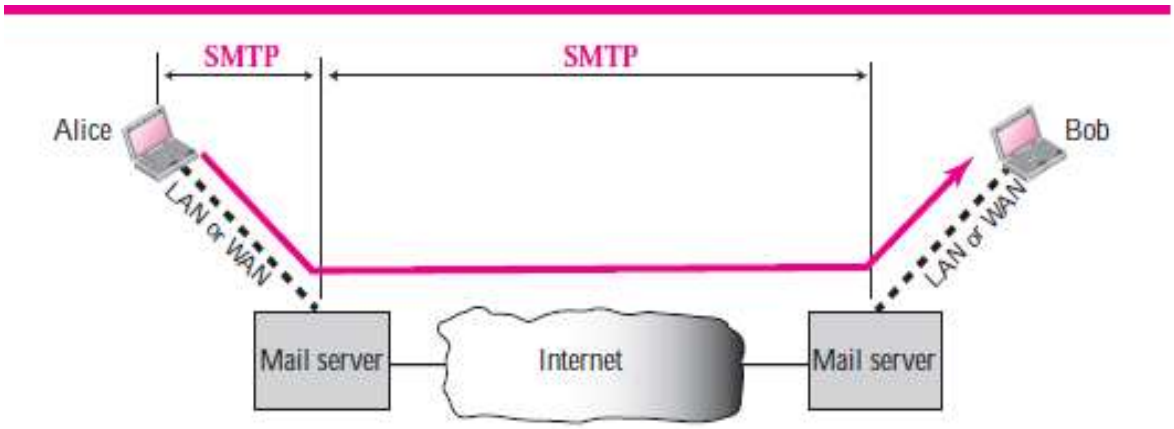
The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called *mail servers* or *exchangers*. The domain name assigned to each mail exchanger either comes from
the DNS database or is a logical name (for example, the name of the organization).

Mailing List or Group List

Electronic mail allows one name, an **alias,** to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA. If there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity

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).** As we said before, two pairs of MTA clientserver programs are used in the most common situation (fourth scenario).
shows the range of the SMTP protocol in this scenario.



SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver. SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We will discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server
HELO. This command is used by the client to identify itself. The argument is the domain name of the client host. The format is

- **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

❑ **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

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

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

❑ **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

❑ **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

❑ **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

❑ **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

❑ **HELP.** This command asks the recipient to send information about the command sent as the argument. The format is

❑ **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

❑ **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

❑ **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 th sender. The format is

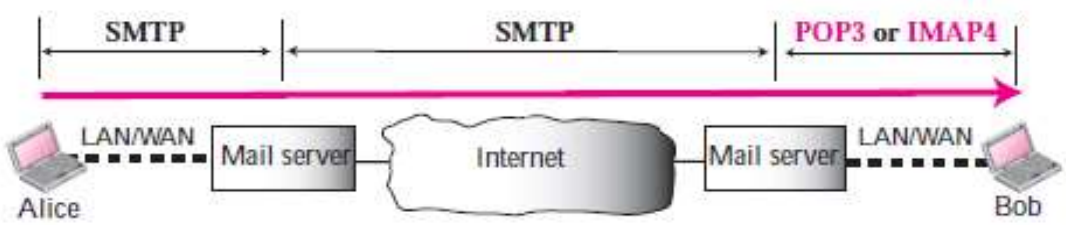
Responses

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

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.

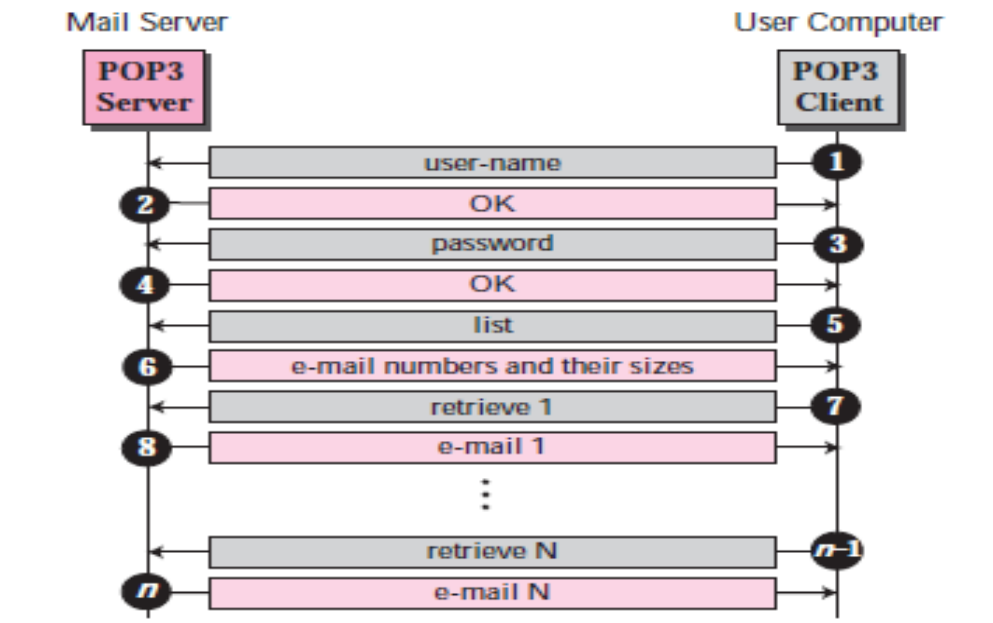
Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).



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 23.14 shows an example of downloading using POP3.

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:

- ❑ A user can check the e-mail header prior to downloading.
- ❑ A user can search the contents of the e-mail for a specific string of characters prio to downloading.
- ❑ A user can partially download e-mail. This is especially useful if bandwidth is limited
and the e-mail contains multimedia with high bandwidth requirements.
- ❑ A user can create, delete, or rename mailboxes on the mail server.
- ❑ A user can create a hierarchy of mailboxes in a folder for e-mail storage

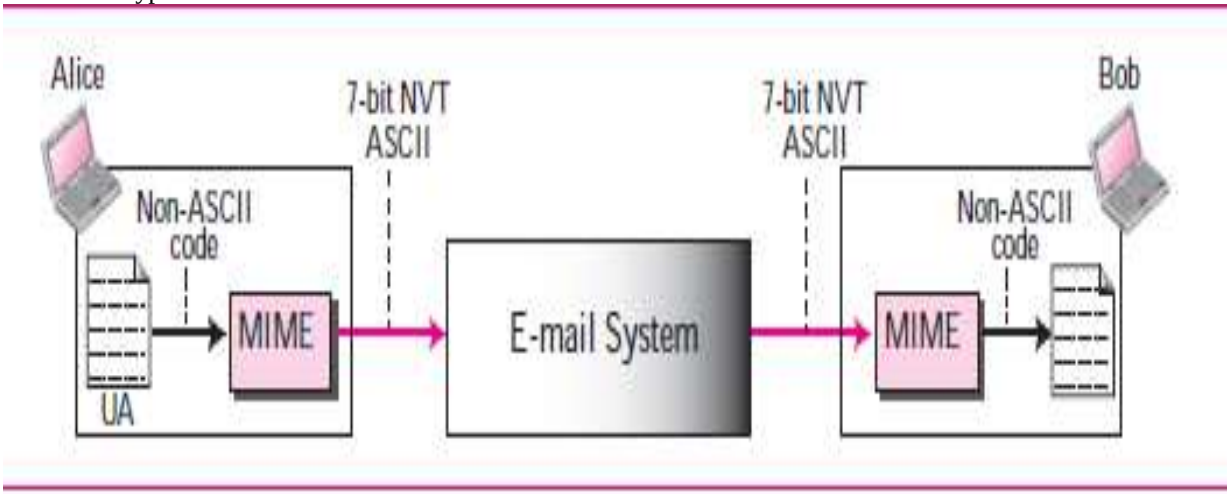
MIME

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,

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

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

- **Text.** The original message is in 7-bit ASCII format and no transformation by MIME is needed. There are two subtypes currently used, *plain* and *HTML*.
- **Multipart.** The body contains multiple, independent parts. The multipart header needs to define the boundary between each part. A parameter is used for this purpose. The parameter is a string token that comes before each part; it is on a separate line
by itself and is preceded by two hyphens. The body is terminated using the boundary token, again preceded by two hyphens and then terminated with two hyphens. Four subtypes are defined for this type: *mixed*, *parallel*, *digest*, and *alternative*. In the mixed subtype, the parts must be presented to the recipient in the exact order as in the message. Each part has a different type and is defined at the boundary. The parallel subtype is similar to the mixed subtype, except that the order of the parts is unimportant. The digest subtype is also similar to the mixed subtype except that the default type/subtype is message/RFC822 as defined below. In the alternative subtype, the same message is repeated using different formats. The following is an
example of a multipart message using a mixed subtype:

❑ **Message.** In the message type, the body is itself an entire mail message, a part of a mail message, or a pointer to a message. Three subtypes are currently used: *RFC822, partial,* and *external-body.* The subtype RFC822 is used if the body is
encapsulating another message (including header and the body). The partial subtype is used if the original message has been fragmented into different mail messages and this mail message is one of the fragments. The fragments must be
reassembled at the destination by MIME. Three parameters must be added: *id, number,* and the *total*. The id identifies the message and is present in all the fragments. The number defines the sequence order of the fragment. The total defines
the number of fragments that comprise the original message

❑ **Image.** The original message is a stationary image, indicating that there is no animation. The two currently used subtypes are *Joint Photographic Experts Group (JPEG),* which uses image compression, and *Graphics Interchange Format (GIF).*

❑ **Video.** The original message is a time-varying image (animation). The only subtype is *Moving Picture Experts Group* (*MPEG*). If the animated image contains sounds, it must be sent separately using the audio content type.

❑ **Audio.** The original message is sound. The only subtype is basic, which uses 8-kHz
standard audio data.

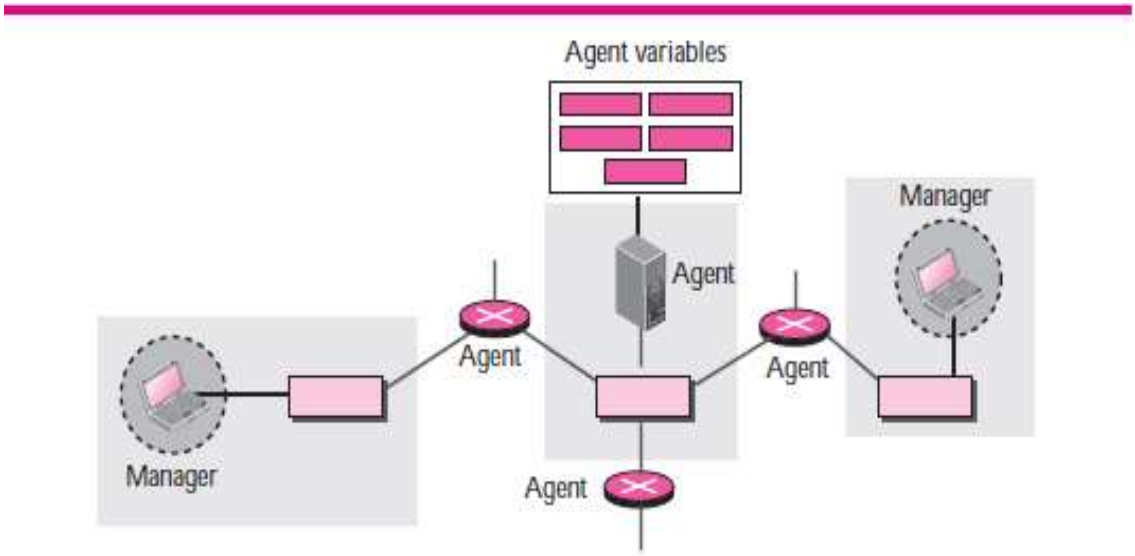
❑ **Application.** The original message is a type of data not previously defined. There are only two subtypes used currently: *PostScript* and *octet-stream.* PostScript is used when the data are in Adobe PostScript format. Octet-stream is used when the data must be interpreted as a sequence of 8-bit bytes (binary file).

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



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.

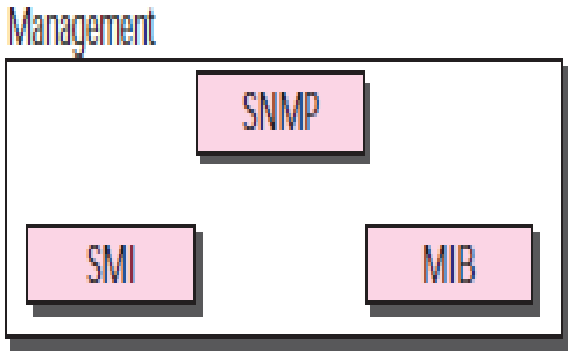
Managers and Agents

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,



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? We need these universal rules because we do not know the architecture of the computers that send, receive, or store these values. The sender may be a powerful computer
in which an integer is stored as 8-byte data; the receiver may be a small computer that stores an integer as 4-byte data.
SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type. SMI is a collection of general rules to name objects and
to list their types. The association of an object with the type is not done by SMI.

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).

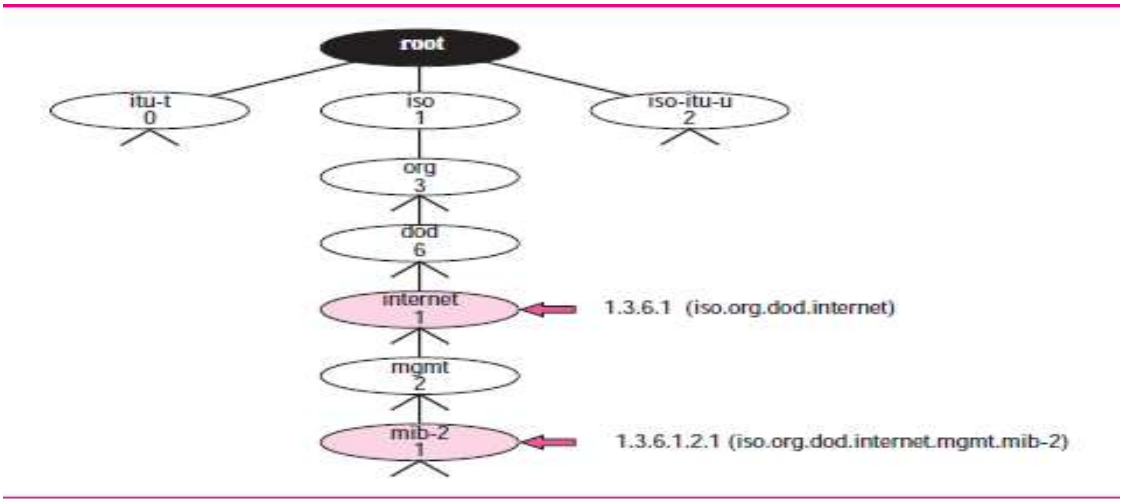
SMI

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



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 Table 24.1. The first five are from ASN.1; the next seven are defined by SMI.

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.

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,

SNMP

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

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

PDUs

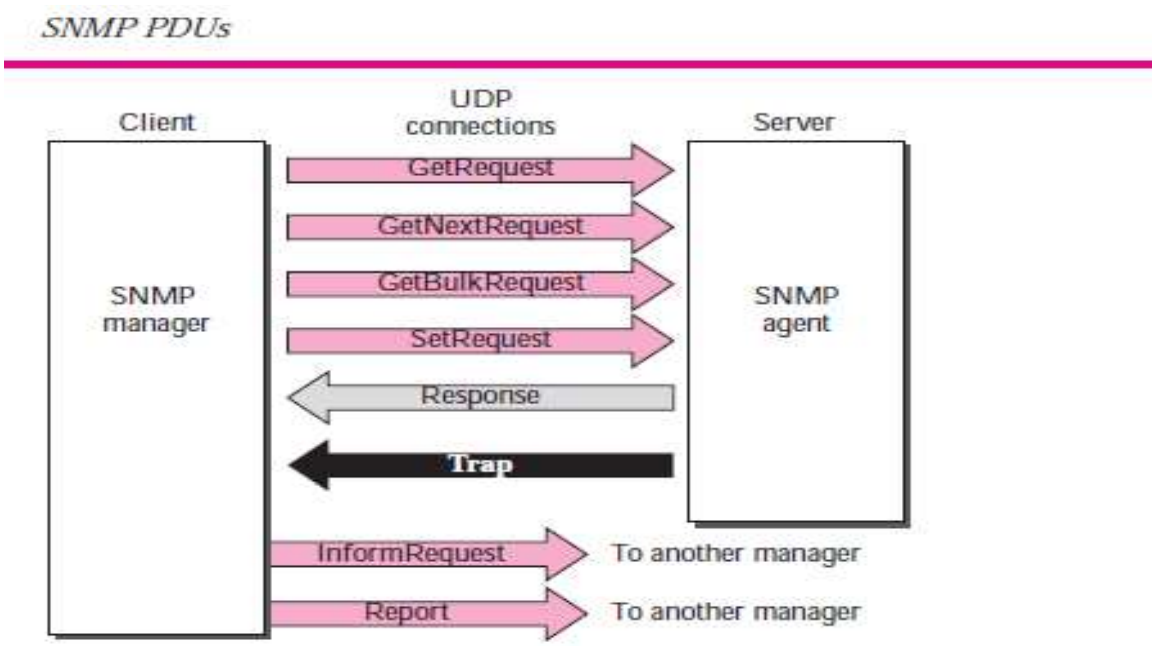
SNMPv3 defines eight types of protocol data units (or PDUs): GetRequest, GetNext- Request, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report.

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.

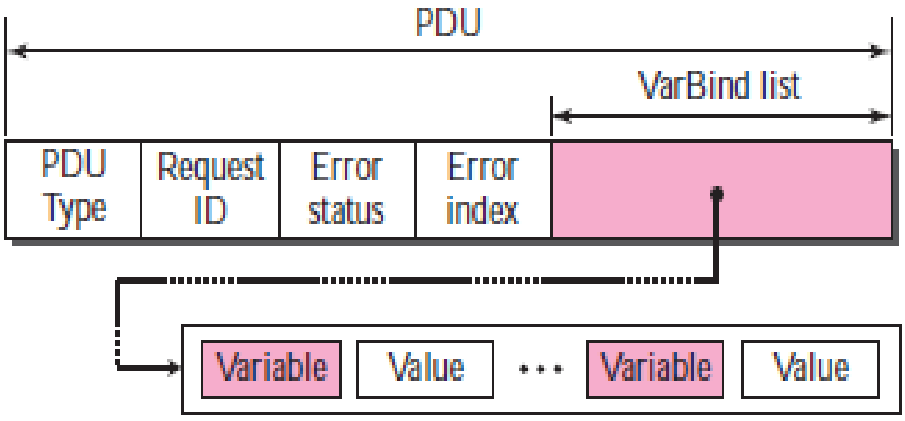
Report

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

Format

The GetBulkRequest PDU differs from the others in two areas

SNMP PDU format



The fields are listed below:

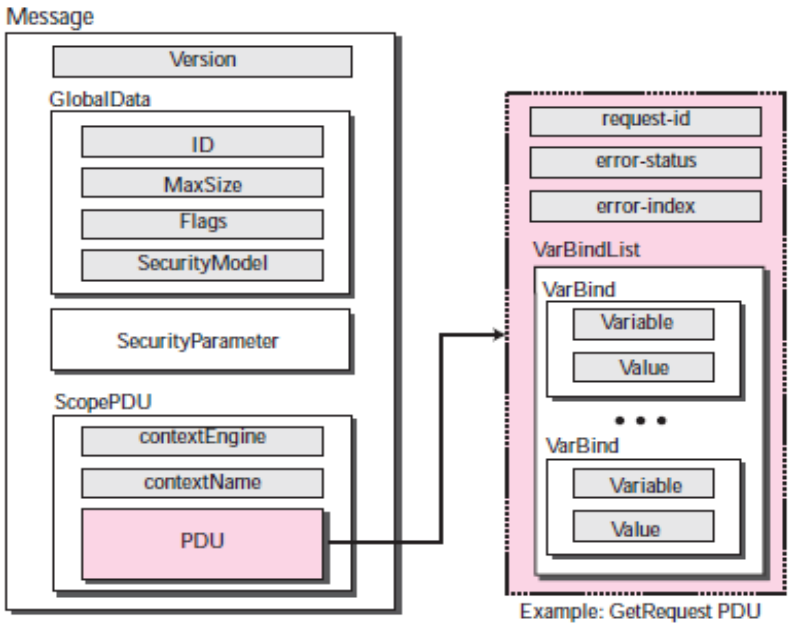
- **PDU type.** This field defines the type of the PDU

Type	*Tag (Binary)*	*Tag (Hex)*
GetRequest	10100000	A0
GetNextRequest	10100001	A1
Response	10100010	A2
SetRequest	10100011	A3
GetBulkRequest	10100101	A5
InformRequest	10100110	A6
Trap (SNMPv2)	10100111	A7
Report	10101000	A8

- **Request ID.** This field is a sequence number used by the manager in a request PDU and repeated by the agent in a response. It is used to match a request to a response.
- **Error status.** This is an integer that is used only in response PDUs to show the types of errors reported by the agent. Its value is 0 in request PDUs. Table lists the types of errors that can occur.
- **Non-repeaters.** This field is used only in GetBulkRequest and replaces the error status field, which is empty in request PDUs.
- **Error index.** The error index is an offset that tells the manager which variable caused the error.
- **Max-repetition.** This field is also used only in GetBulkRequest and replaces the error index field, which is empty in request PDUs.
- **VarBind list.** This is a set of variables with the corresponding values the manager wants to retrieve or set. The values are null in GetRequest and GetNextRequest. In a Trap PDU, it shows the variables and values related to a specific PDU.

Messages

SNMP does not send only a PDU, it embeds the PDU in a message. A message in SNMPv3 is a sequence made of four elements: Version, GlobalData, SecurityParameters, and ScopePDU (which includes the encoded PDU. The first and the third elements are simple data types; the second and the fourth are sequences.



Version

The Version field is an INTEGER data type that defines the version. The current version is 3.

GlobalData

The GlobalData field is a sequence with four elements of simple data type: ID, Max- Size, Flags, and SecurityModel.

Security Parameter

This element is a sequence that can be very complex, depending on the type of security provision used in version 3.

ScopePDU

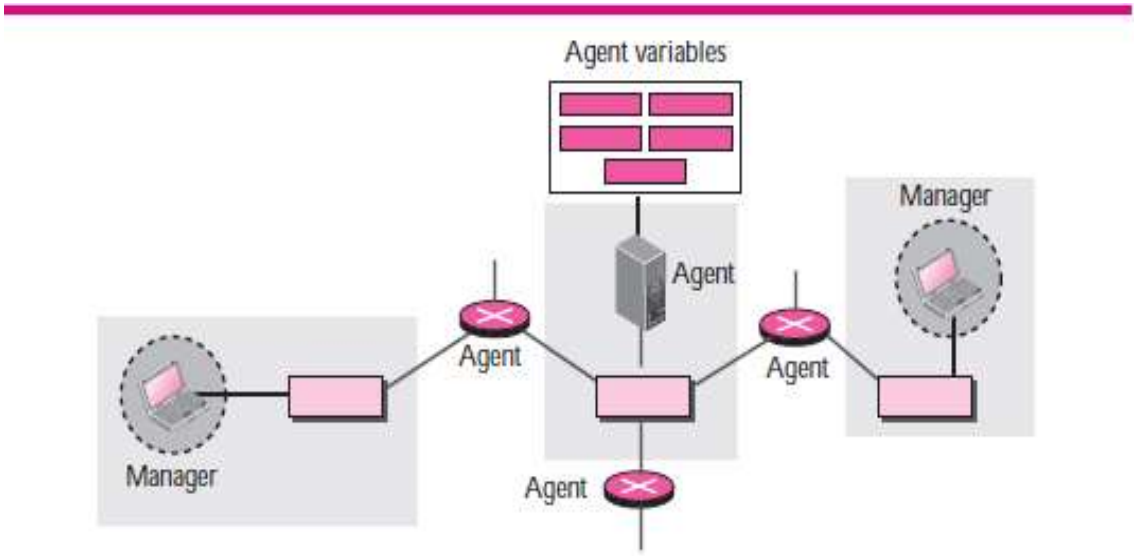
The last element contains two simple data type and the actual PDU. We have shown only one example of GetRequest PDU. Note that VarBindList is a sequence made of one or more sequences called VarBind. Each VarBind is made of two simple data elements: Variable and Value.

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



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.

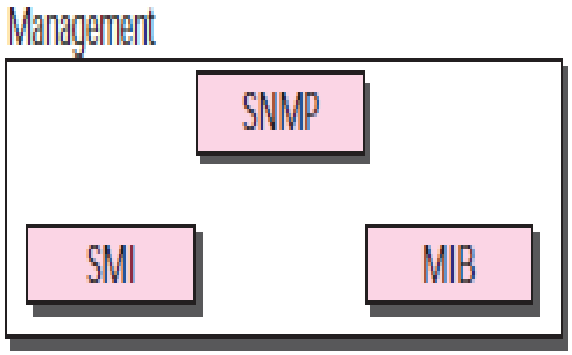
Managers and Agents

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,



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? We need these universal rules because we do not know the architecture of the computers that send, receive, or store these values. The sender may be a powerful computer
in which an integer is stored as 8-byte data; the receiver may be a small computer that stores an integer as 4-byte data.
SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type. SMI is a collection of general rules to name objects and
to list their types. The association of an object with the type is not done by SMI.

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).

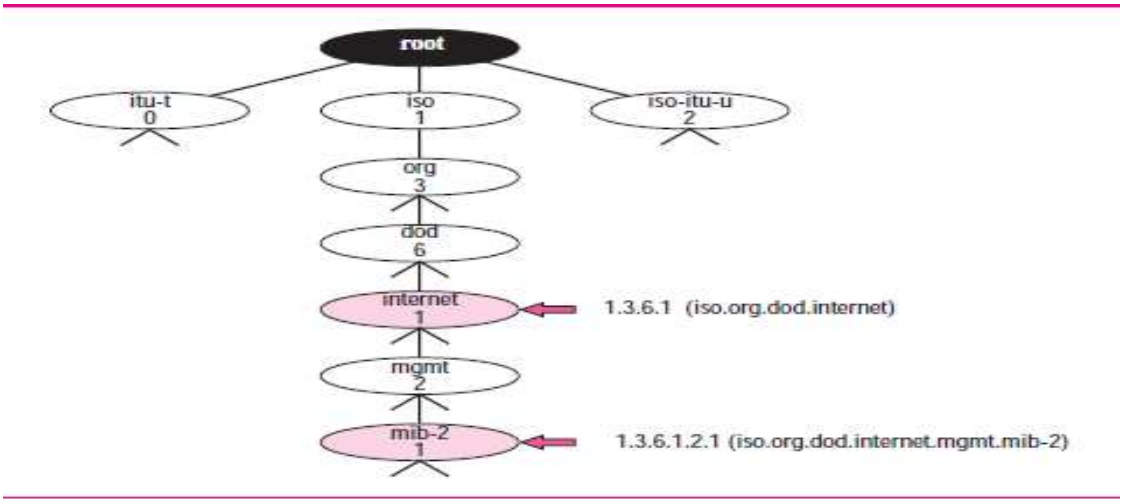
SMI

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



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 Table 24.1. The first five are from ASN.1; the next seven are defined by SMI.

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.

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,

SNMP

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

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

PDUs

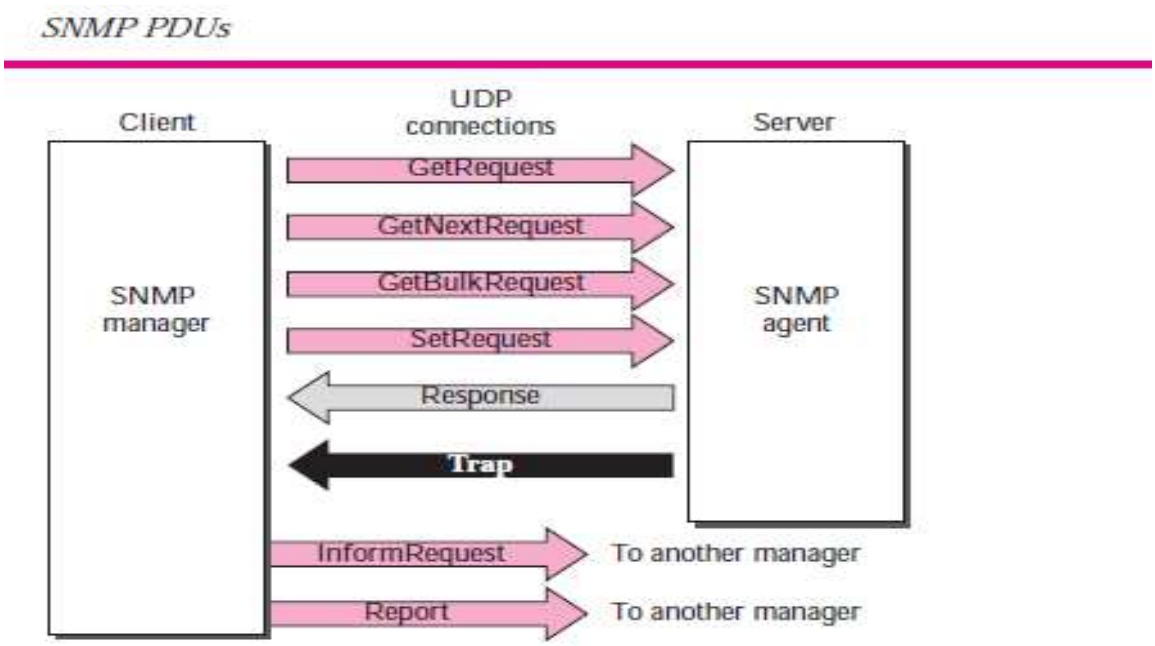
SNMPv3 defines eight types of protocol data units (or PDUs): GetRequest, GetNext- Request, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report.

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.

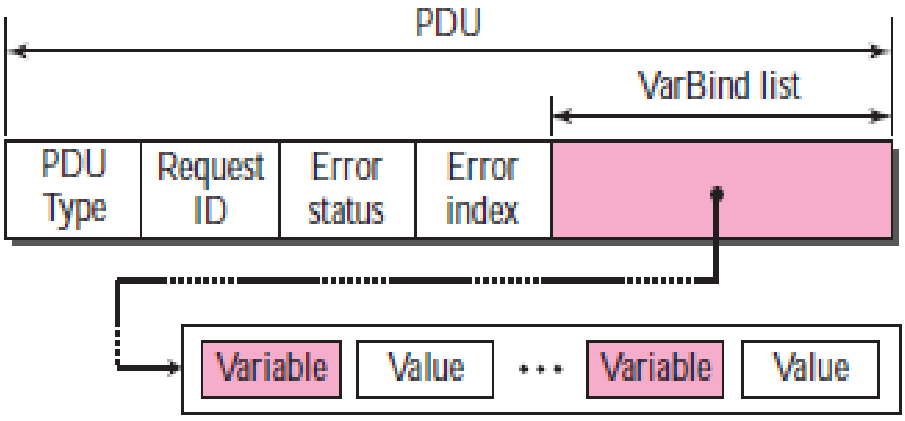
Report

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

Format

The GetBulkRequest PDU differs from the others in two areas

SNMP PDU format



The fields are listed below:

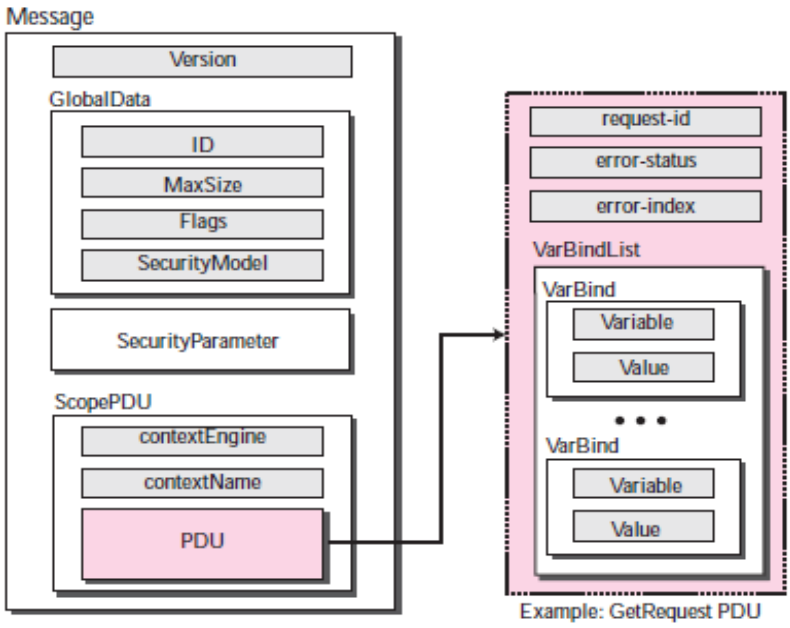
- **PDU type.** This field defines the type of the PDU

Type	*Tag (Binary)*	*Tag (Hex)*
GetRequest	10100000	A0
GetNextRequest	10100001	A1
Response	10100010	A2
SetRequest	10100011	A3
GetBulkRequest	10100101	A5
InformRequest	10100110	A6
Trap (SNMPv2)	10100111	A7
Report	10101000	A8

- **Request ID.** This field is a sequence number used by the manager in a request PDU and repeated by the agent in a response. It is used to match a request to a response.
- **Error status.** This is an integer that is used only in response PDUs to show the types of errors reported by the agent. Its value is 0 in request PDUs. Table lists the types of errors that can occur.
- **Non-repeaters.** This field is used only in GetBulkRequest and replaces the error status field, which is empty in request PDUs.
- **Error index.** The error index is an offset that tells the manager which variable caused the error.
- **Max-repetition.** This field is also used only in GetBulkRequest and replaces the error index field, which is empty in request PDUs.
- **VarBind list.** This is a set of variables with the corresponding values the manager wants to retrieve or set. The values are null in GetRequest and GetNextRequest. In a Trap PDU, it shows the variables and values related to a specific PDU.

Messages

SNMP does not send only a PDU, it embeds the PDU in a message. A message in SNMPv3 is a sequence made of four elements: Version, GlobalData, SecurityParameters, and ScopePDU (which includes the encoded PDU. The first and the third elements are simple data types; the second and the fourth are sequences.



Version

The Version field is an INTEGER data type that defines the version. The current version is 3.

GlobalData

The GlobalData field is a sequence with four elements of simple data type: ID, Max- Size, Flags, and SecurityModel.

Security Parameter

This element is a sequence that can be very complex, depending on the type of security provision used in version 3.

ScopePDU

The last element contains two simple data type and the actual PDU. We have shown only one example of GetRequest PDU. Note that VarBindList is a sequence made of one or more sequences called VarBind. Each VarBind is made of two simple data elements: Variable and Value.

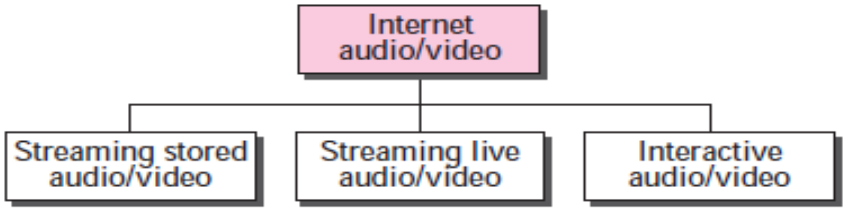
Multimedia

Recent advances in technology have changed our use of audio and video. In the past, we listened to an audio broadcast through a radio and watched a video program broadcast through a TV. We used the telephone network to interactively communicate with another party. But times have changed. People want to use the Internet not only for text and image communications, but also for audio and video services. In this chapter, we concentrate on applications that use the Internet for audio and video services.

INTRODUCTION

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

Internet audio/video



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

Before audio or video signals can be sent on the Internet, they need to be digitized. We discuss audio and video separately.

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 $2f$ 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 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.** In this section, we first discuss audio compression and then video 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). Detailed discussions of these techniques are beyond the scope of this book.

Perceptual Encoding: MP3

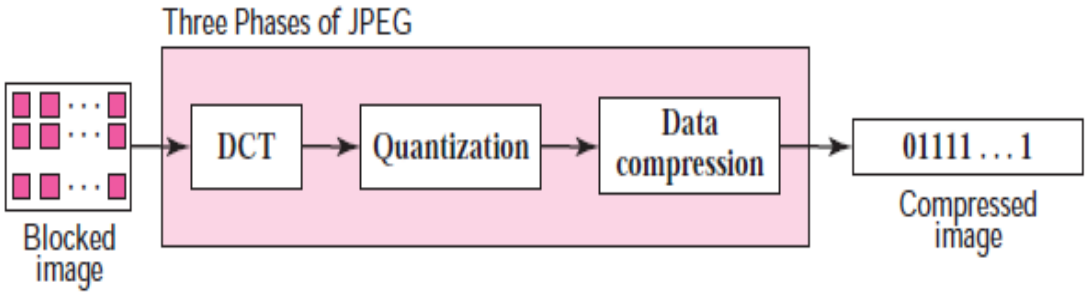
The most common compression technique that is used to create CD-quality audio is based on the **perceptual encoding** technique. As we mentioned before, this type of audio needs at least 1.411 Mbps; this cannot be sent over the Internet without compression. **MP3** (MPEG audio layer 3), a part of the 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 softer 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

As we mentioned before, 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. We briefly discuss JPEG and then MPEG.

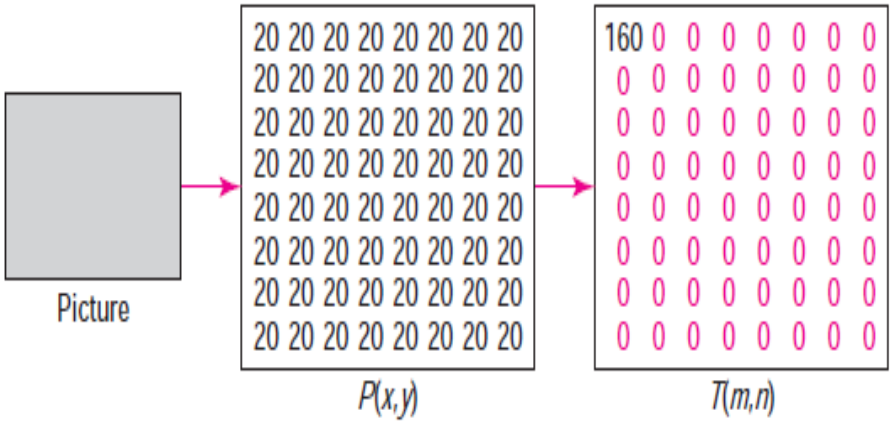
Image Compression: JPEG

As we discussed previously, 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 The purpose of dividing the picture into blocks is to decrease the number of calculations
because, as you will see shortly, 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



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. We do not give the formula here, but we do show the results
of the transformation for three cases.

Case 1 In this case, we have a block of uniform gray, and the value of each pixel is 20. When we do the transformations, we get a nonzero value for the first element (upper left corner); the rest of the pixels have a value of 0. The value of $T(0,0)$ is the
average (multiplied by a constant) of the $P(x,y)$ values and is called the *dc value* (direct current, borrowed from electrical engineering). The rest of the values, called *ac values*, in $T(m,n)$ represent changes in the pixel values. But because there are no changes, the
rest of the values are 0s



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. W 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.

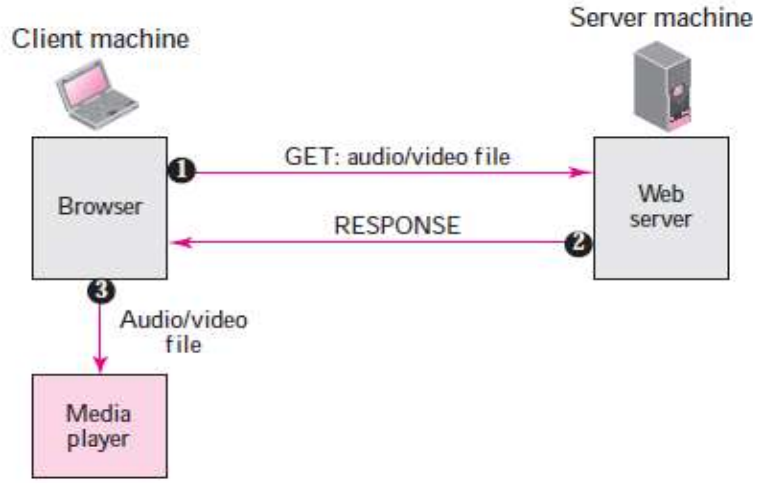
STREAMING STORED AUDIO/VIDEO

Now that we have discussed digitizing and compressing audio/video, we turn our attention to specific applications. 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. This approach is very simple and does not involve *streaming*. However, it has adrawback. 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.

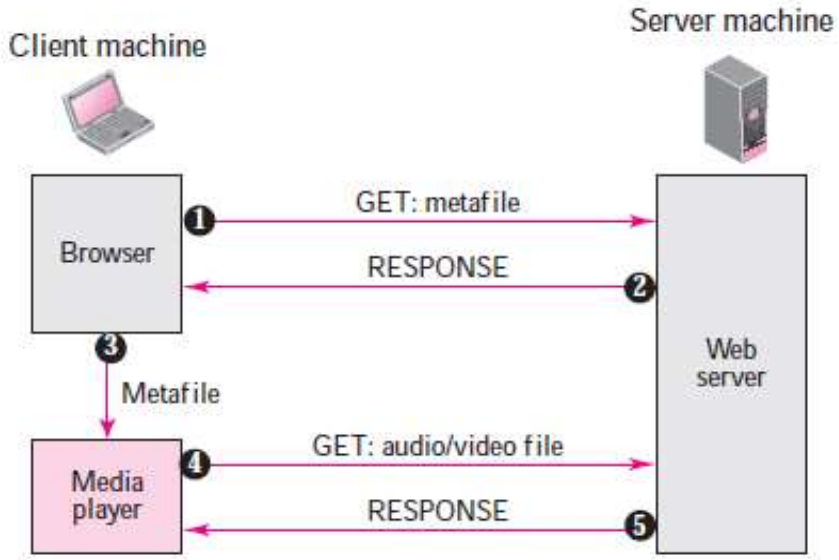
Using a Web server



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.

Using a Web server with a metafile

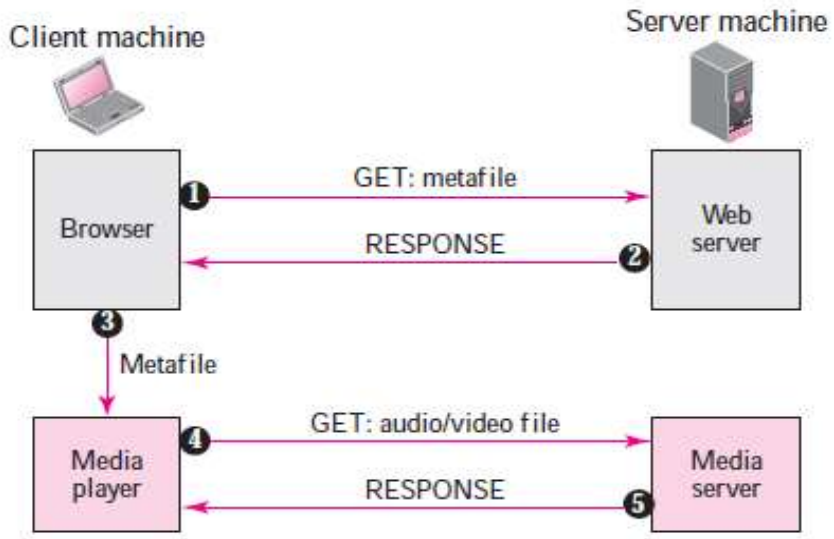


1. The HTTP client accesses the Web server using the GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the audio/video file.
5. The Web server responds.

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.**

Using a media server



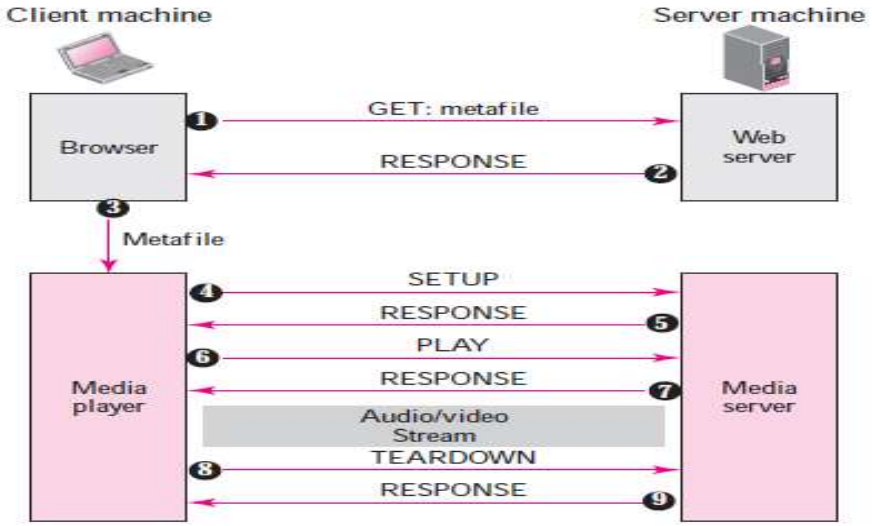
1. The HTTP client accesses the Web server using a GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the media server to download the file. Downloading can take place by any 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 25.13 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.

Using a media server and RTSP



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.

STREAMING LIVE AUDIO/VIDEO

Streaming live audio/video is similar to the broadcasting of audio and video by radio and TV stations. Instead of broadcasting to the air, the stations broadcast through the Internet. There are several similarities between streaming stored audio/video and streaming live audio/video. They are both sensitive to delay; neither can accept retransmission. However, there is a difference. In the first application, the communication is unicast and on-demand. In the second, the communication is multicast and live. Live
streaming is better suited to the multicast services of IP and the use of protocols such as UDP and RTP (discussed later). However, presently, live streaming is still using TCP and multiple unicasting instead of multicasting. There is still much progress to be made in this area.

REAL-TIME INTERACTIVE AUDIO/VIDEO

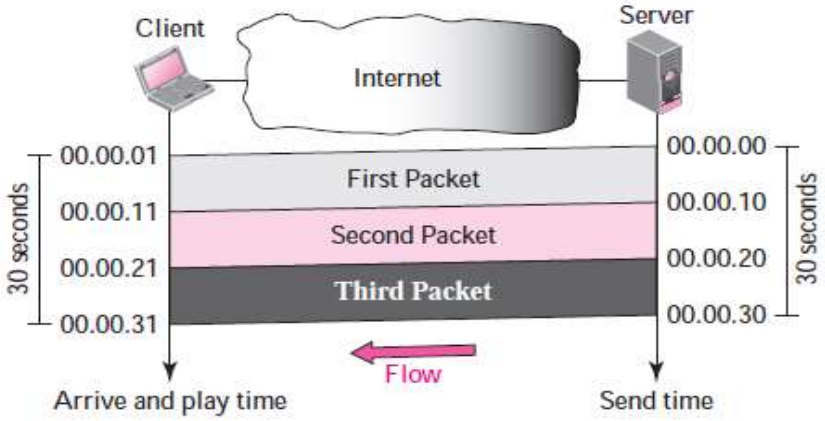
In real-time interactive audio/video, people communicate with one another in real time. The Internet phone or voice over IP is an example of this type of application. Video conferencing is another example that allows people to communicate visually and orally.

Characteristics

Before discussing the protocols used in this class of applications, we discuss 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. For example, let us assume that a real-time video server creates live video images and sends them online. The video is digitized and packetized. There are only three packets, and each packet holds 10 s of video information. The first packet starts at 00:00:00, the second packet starts at 00:00:10, and the third packet starts at 00:00:20. Also imagine that it takes 1 s (an exaggeration for simplicity) for each packet to reach the destination (equal delay). The receiver can play back the first packet at 00:00:01, the second packet at 00:00:11, and the third packet at 00:00:21. Although there is a 1-s time difference between what the server sends and what the client sees on the computer screen, the action is happening in real time. The time relationship between the packets is preserved. The 1-s delay is not important.



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. In other words, the
receiver knows when each packet is to be played. Imagine the first packet in the previous example has a timestamp of 0, the second has a timestamp of 10, and the third a timestamp of 20. If the receiver starts playing back the first packet at 00:00:08, the second will be played at 00:00:18, and the third at 00:00:28. There are no gaps between the packets.

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. In the previous example, the first bit of the first packet arrives at 00:00:01; the threshold is 7 s, and the playback time is 00:00:08. The threshold is measured in time units of data. The replay does not start until the time units of data are equal to the threshold value. Data are stored in the buffer at a possibly variable rate, but they are extracted and played back at a fixed rate. Note that the amount of data in the buffer shrinks or expands, but as long as the delay is less than the time to play back the threshold amount of data, there is no jitter. Figure 25.17 shows the buffer at different times for ou example.

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. For example, suppose the timestamps
are 0, 10, and 20. If the second packet is lost, the receiver receives just two packets with timestamps 0 and 20. The receiver assumes that the packet with timestamp 20 is the second packet, produced 20 s after the first. 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 ca change the format of a high-bandwidth video signal to a lower-quality narrowbandwidth signal. This is needed, for example, for a source creating a high-quality video
signal at 5 Mbps and sending to a recipient having a bandwidth of less than 1 Mbps. T receive the signal, a translator is needed to decode the signal and encode it again at a lower quality that needs less bandwidth.

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 on stream, data from different sources can be mixed. A **mixer** mathematically adds signals coming from different sources to create one single signal.

Support from Transport Layer Protocol

The procedures mentioned in the previous sections can be implemented in the application layer. However, they are so common in real-time applications that implementation in the transport layer protocol is preferable. Let's see which of the existing transport layers is suitable for this type of traffic.

TCP is not suitable for interactive traffic. It has no provision for timestamping, and it does not support multicasting. However, it does provide ordering (sequence numbers). One feature of TCP that makes it particularly unsuitable for interactive traffic is

its error control mechanism. In interactive traffic, we cannot allow the retransmission of a lost or corrupted packet. If a packet is lost or corrupted in interactive traffic, it must just be ignored. Retransmission upsets the whole idea of timestamping and playback. Today there is so much redundancy in audio and video signals (even with compression) that we can simply ignore a lost packet. The listener or viewer at the remote site may not even notice it. UDP is more suitable for interactive multimedia traffic. UDP supports multicasting and has no retransmission strategy. However, UDP has no provision for timestamping,

sequencing, or mixing. A new transport protocol, Real-Time Transport Protocol (RTP), provides these missing features.

QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue that has been discussed more tha defined. We can informally define quality of service as something a flow of data seeks to attain. Although QoS can be applied to both textual data and multimedia, it is more an issue when we are dealing with multimedia.

Flow Characteristics

Traditionally, four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth

Reliability

Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. However, the sensitivity of application programs to reliability is not the same. For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay

Source-to-destination **delay** is another flow characteristic. Again applications can tolerate delay in different degrees. In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

Jitter

Jitter is the variation in delay for packets belonging to the same flow. For example, if four packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time. On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21, 22, 19, and 24. For applications such as audio and video, the first case is completely acceptable; the second case is not. For these applications, it does not matter if the packets arrive

with a short or long delay as long as the delay is the same for all packets. For this application, the second case is not acceptable.

Jitter is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.

Bandwidth

Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

Flow Classes

Based on the flow characteristics, we can classify flows into groups, with each group having similar levels of characteristics. This categorization is not formal or universal; some protocols such as ATM have defined classes.

Techniques to Improve QoS

In the previous section we tried to define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.

Scheduling

Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. We discuss three of them here: FIFO queuing, priority queuing, and weighted fair queuing.

FIFO Queuing In **first-in, first-out (FIFO) queuing,** packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will
be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bu stop
Priority Queuing In **priority queuing,** packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving a queue until it is empty. A priority queue can provide better QoS than the FIFO queue because higherpriority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called *starvation*.
Weighted Fair Queuing A better scheduling method is **weighted fair queuing.** In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, however, are weighted based on the priority of the queues;
higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.

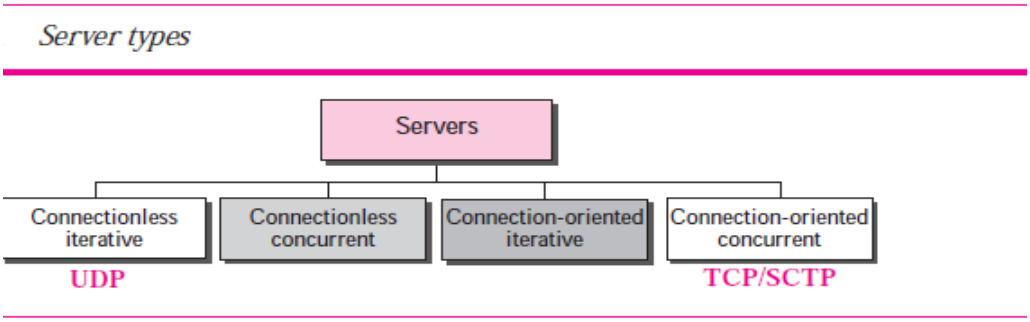
Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.
Leaky Bucket If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called **leaky bucket** can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate

Token Bucket The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the **token bucket** algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends *n* tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if *n* is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1,000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty

Combining Token Bucket and Leaky Bucket The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the
rate of tokens dropped in the bucket.

UNIT :- VI



Difference Between Connection-oriented and Connection-less Services

Communication can be established in two ways between two or more devices that are connection-oriented and connection-less. Network layers can offer these two different types of services to its predecessor layer for transferring data. **Connection-oriented services** involve the establishment and termination of the connection while **connection-less services** don't require any connection creation and termination processes for transferring data.

Another difference between connection-oriented and connection-less services is connection-oriented communication uses a stream of data and is vulnerable to router failure while connection-less communication uses messages and is robust to router failure.

Content: Connection-oriented Vs Connection-less Services

Comparison Chart

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE
Prior Connection Requirement	Necessary	Not required
Reliability	Ensures reliable transfer of data.	Not guaranteed.
Congestion	Unlikely	Occur likely.
Transferring mode	It can be implemented using circuit switching and virtual circuit.	It is implemented using packet switching.
Lost data retransmission	Feasible	Practically, not possible.
Suitability	Suitable for long and steady communication.	Suitable for bursty Transmission.
Signalling	Used for connection establishment.	There is no concept of signalling.

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE
Packet forwarding	Packets sequentially travel to their destination node and follows the same route.	Packets reach the destination randomly without following the same route.
Delay	There is a delay in transfer of information, but once the connection is established faster delivery can be achieved.	Because to the absence of connection establishment phase, the transmission is faster.
Resource Allocation	Need to be allocated.	No prior allocation of the resource is required.

Definition of Connection-oriented Service

Connection-oriented service is analogous to the **telephone system** that requires communication entities to establish a connection before sending data. TCP provides Connection-oriented services as does **ATM, Frame Relay** and **MPLS** hardware. It uses **handshake process** to establish the connection between the sender and receiver.

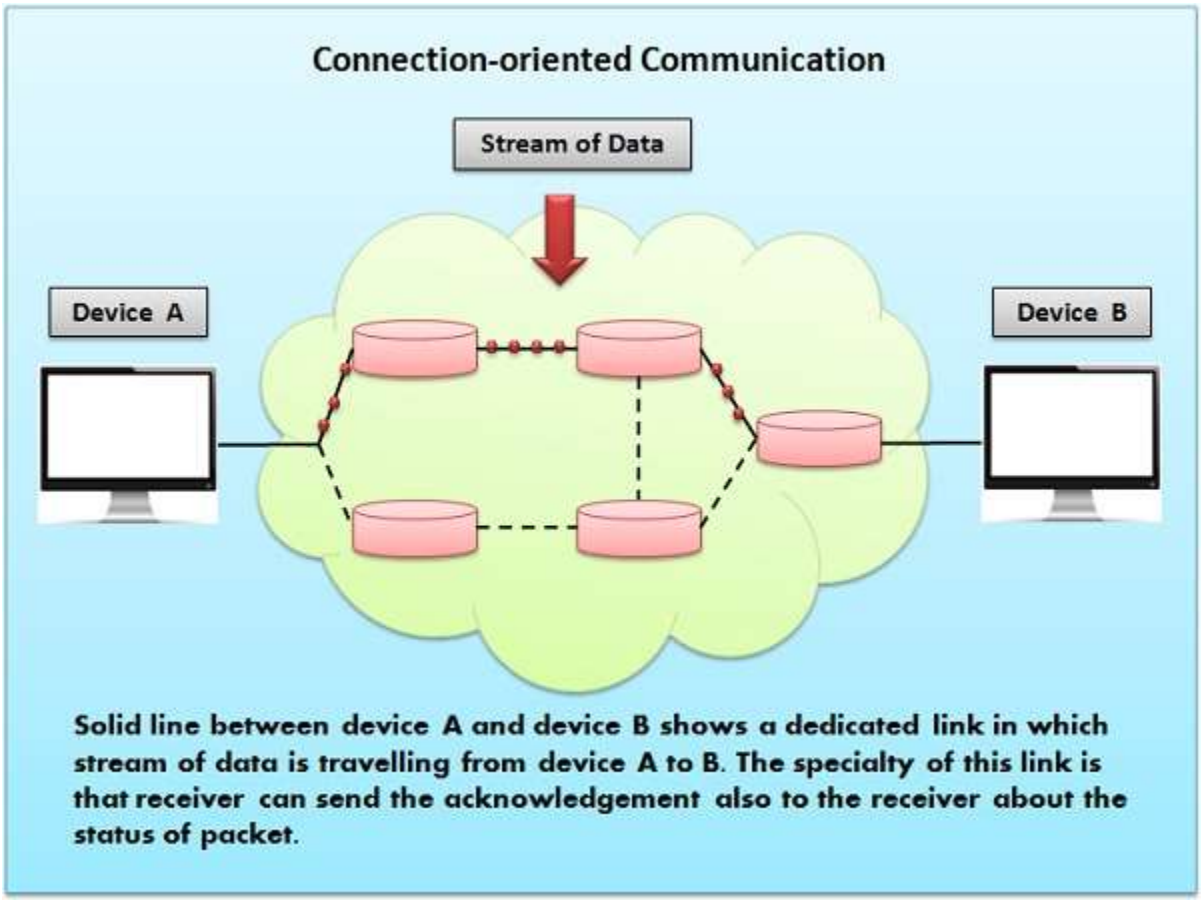
A handshake process includes some steps which are:

- Client requests server to set up a connection for transfer of data.

- Server program notifies its TCP that connection can be accepted.
- The client transmits a SYN segment to the server.
- The server sends SYN+ACK to the client.
- Client transmits 3rd segment i.e. just ACK segment.
- Then server terminates the connection.

More precisely, it sets up a connection uses that connection then terminates the connection.

Reliability is achieved by having recipient acknowledge each message. There are **sequencing** and **flow control,** that's the reason packets received at the receiving end are always in **order**. It uses **circuit switching** for transmission of data.



Connection-oriented transport service priorly constructs a **virtual circuit**between two remote devices. To this end, COTS makes four different kinds of services available to the upper layers:

T-CONNECT	This service enables a full duplex transport connection on a remote device with a peer function.
T-DATA	This service is used to transfer data, it could provide uncertain service and restricted amount of data but still, it is reliable.
T-EXPEDITED-DATA	This service is also used for transferring data, but it carries a limited amount expedited data up to 16 octets (bytes).
T-DISCONNECT	It is used to terminate the Transport connection and to reject a connection request also.

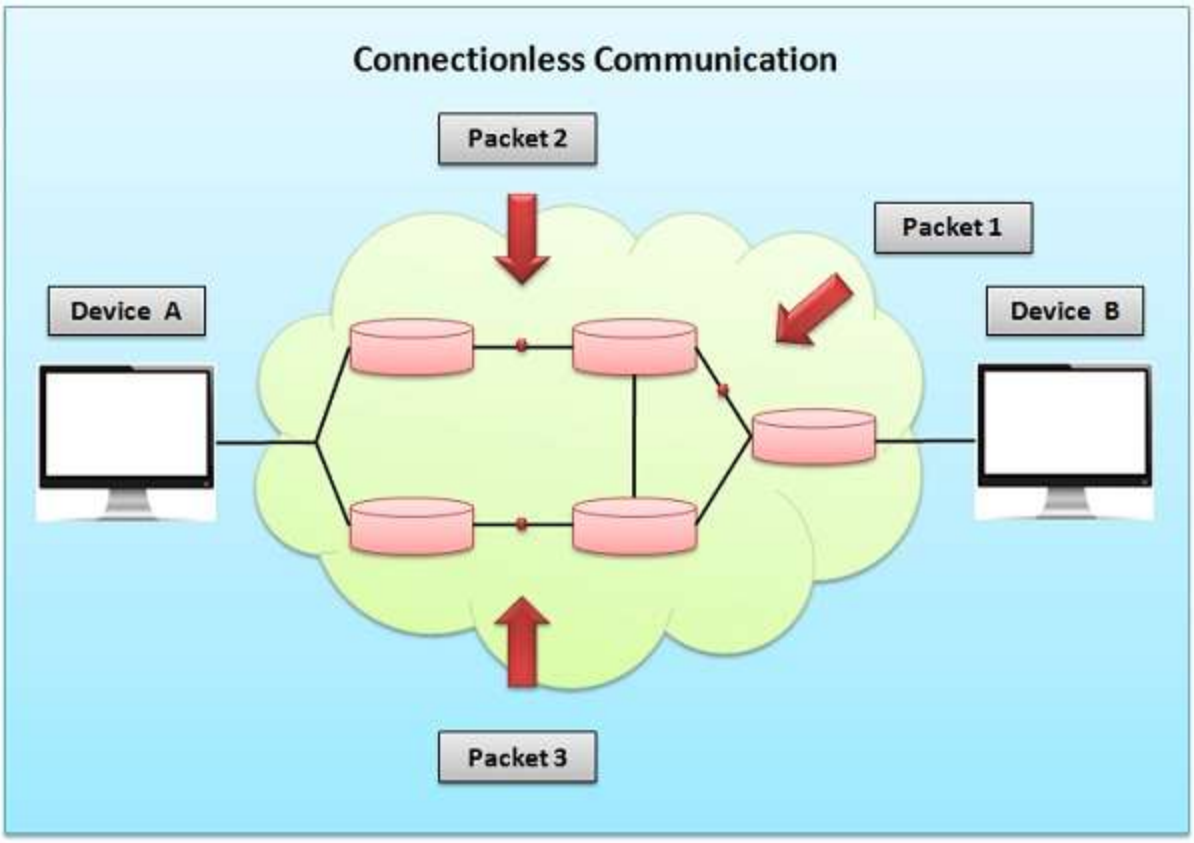
where, T stands for Transfer.

Definition of Connection-less Service

Connection-less service is analogous to the **postal system**. In which packets of data (usually known as a **datagram**) is transmitted from source to destination directly. Each packet is treated as an individual entity, which allows communication entities to send data before establishing communication. Each packet carries a **destination address** to identify the intended recipient.

Packets don’t follow a **fixed path** that is the reason the packets received at receiver end can be out of order. It uses **packet switching** for transmission of data.

Most network hardware, the **Internet Protocol (IP)**, and the **User Datagram Protocol (UDP)** provides connection-less service.



Connection-less Transport services offer only one type of service to its upper layer that is **T-UNIT-DATA**. It provides a single solitary data unit for all transmission. Each unit contains all of the protocol control information necessary for delivery but does not include provision for sequencing and flow control.

Key Differences Between Connection-oriented and Connection-less Services

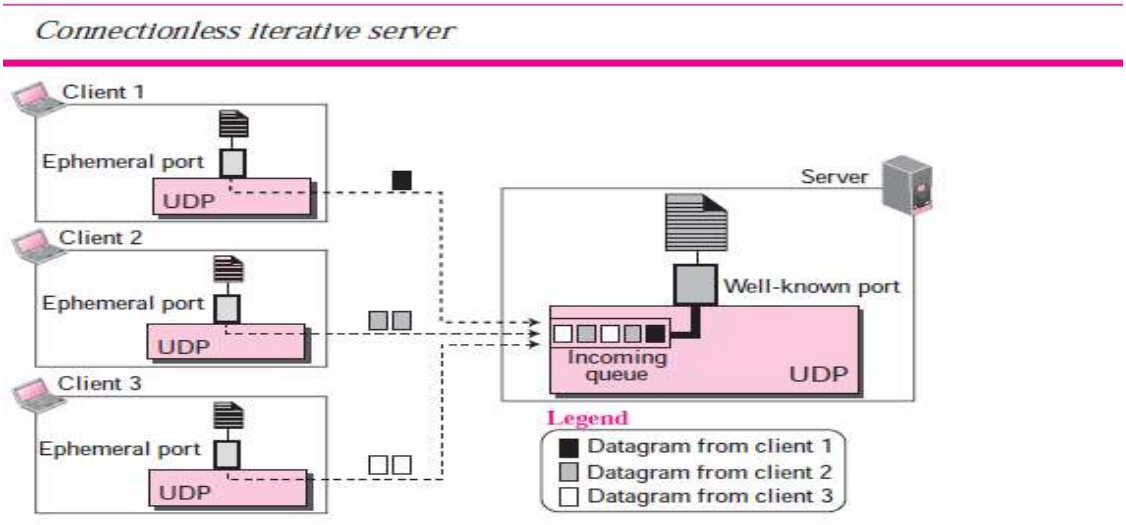
The points given below explains the difference between connection-oriented and connection-less services:

1. There is a requirement for prior connection for communication in connection-oriented services, in contrast, it is not needed in connection-less services.
2. Reliability is more in connection-oriented as compared to connection-less services.
3. Traffic congestion is greater in connection-less services whereas its occurrence is rare in connection-oriented services.
4. In connection-oriented services order of packets received at the destination is same as sent from the source. On the contrary, the order might change in connection-less services.

5. All packets follow the same path in connection-oriented services while packets follow a random path to reach the destination in connection-less services.
6. Connection-oriented service is appropriate for long and steady communication whereas connection-less service is fit for bursty transmission.
7. In connection-oriented services, sender and receiver are synchronized with each other while it is not the case of connection-less services.
8. Connection-oriented services use circuit switching on the other hand packet switching is used in connection-less services.
9. Bandwidth requirement is higher in Connection-oriented services whereas its low in connection-less services.

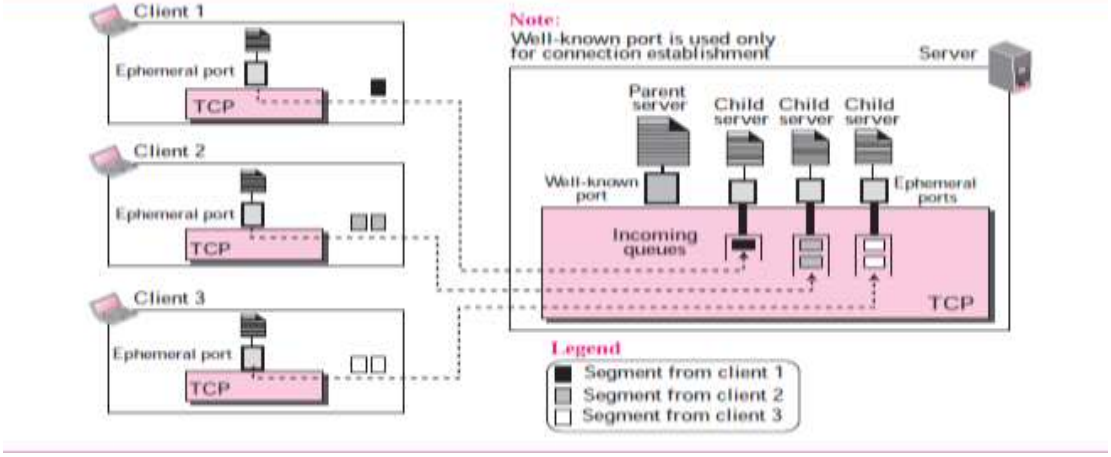
Connectionless Iterative Server

The servers that use UDP are normally iterative, which, as we have said, means that the server processes one request at a time. A server gets the request received in a datagram from UDP, processes the request, and gives the response to UDP to send to the client. The server pays no attention to the other datagrams. These datagrams are stored in a queue, waiting for service. They could all be from one client or from many clients. In either case they are processed one by one in order of arrival. The server uses one single port for this purpose, the well-known port. All the datagrams arriving at this port wait in line to be served



Connection-Oriented Concurrent Server

The servers that use TCP (or SCTP) are normally concurrent. This means that the server can serve many clients at the same time. Communication is connection-oriented, which means that a request is a stream of bytes that can arrive in several segments and the response can occupy several segments. A connection is established between the server and each client, and the connection remains open until the entire stream is processed and the connection is terminated. This type of server cannot use only one port because each connection needs a port and many connections may be open at the same time. Many ports are needed, but a server can use only one well-known port. The solution is to have one well-known port and many ephemeral ports. The server accepts connection requests at the well-known port. A client can make its initial approach to this port to make the connection. After the connection is
made, the server assigns a temporary port to this connection to free the well-known port. Data transfer can now take place between these two temporary ports, one at the client site and the other at the server site. The well-known port is now free for another client to make the connection. To serve several clients at the same time, a server creates child processes, which are copies of the original process (parent process). The server must also have one queue for each connection. The segments come from the client, are stored in the appropriate queue, and will be served concurrently by
the server.



Communication Using UDP

shows a simplified flow diagram for this type of communication.

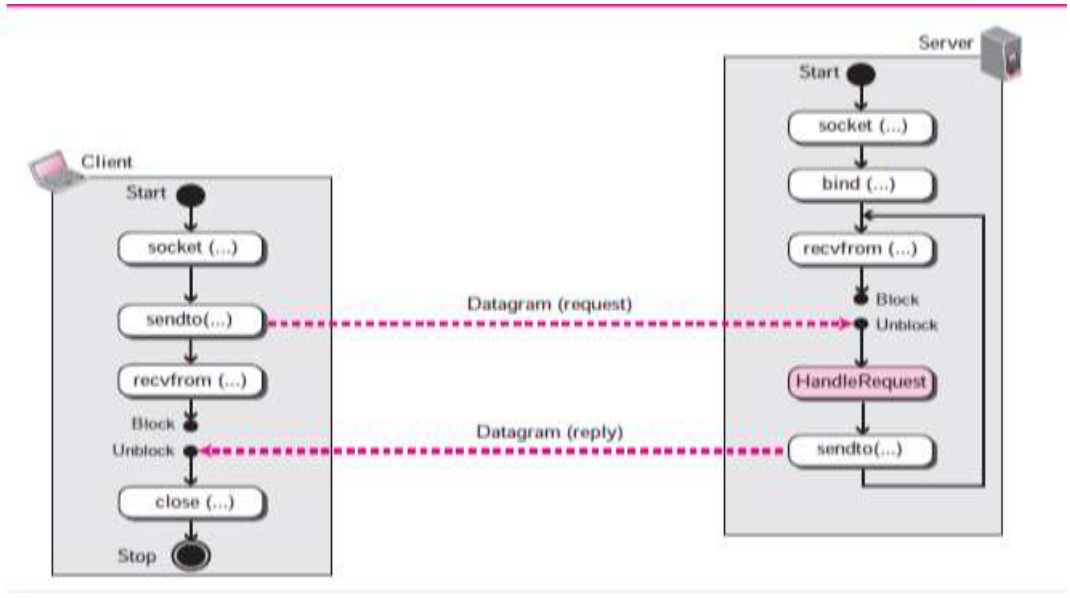


Diagram :-Connection Iterative Connection using UDP

Server Process

The server process starts first. The server process calls the *socket* function to create a socket. It then calls the *bind* function to bind the socket to its well-known port and the IP address of the computer on which the server process is running. The server then calls the *recvfrom* function, which blocks until a datagram arrives. When the datagram arrives, the *recvfrom* function unblocks, extracts the client socket address and address length from the received datagram, and returns them to the process. The process saves these two pieces of information and calls a procedure (function) to handle the request. When the result is ready, the server process calls the *sendto* function and uses the saved information to send the result to the client that requested it. The server uses an infinite loop to respond to the requests coming from the same client or different clients.

Client Process

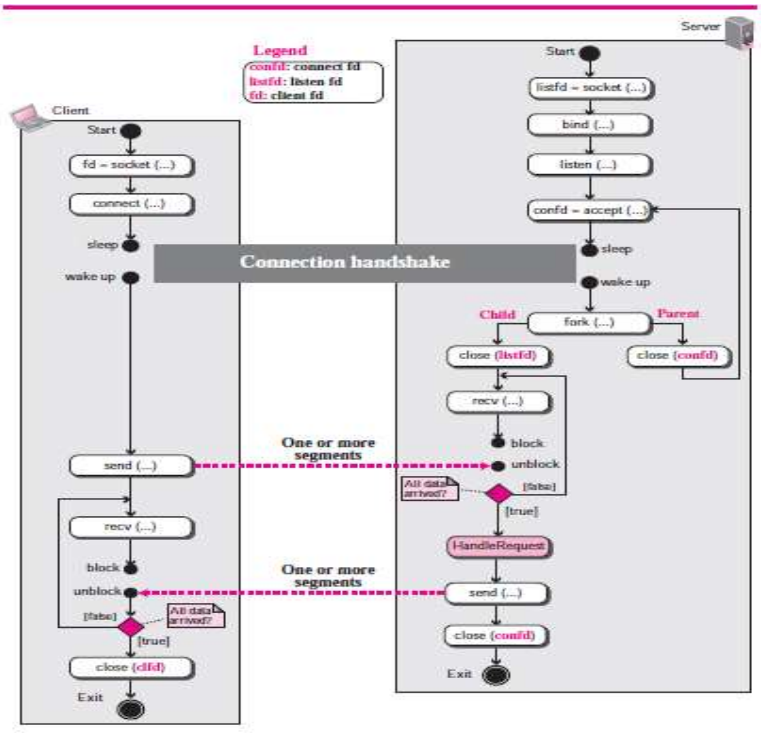
The client process is simpler. The client calls the *socket* function to create a socket. It then calls the *sendto* function and pass the socket address of the server and the location of the buffer from which UDP can get the data to make the datagram. The client then calls a *recvfrom* function call that blocks until the reply arrives from the server. When the reply arrives, UDP delivers the data to the client process, which make the recv function to unblock and deliver the data received to the client process. Note that we assume that the client message is so small that it fits into one single datagram. If this is not the case, the two function calls, *sendto* and *recvfrom*, need to be repeated. However, the server is not aware of multidatagram communication; it handles each request separately

Communication Using TCP

Now we discuss connection-oriented, concurrent communication using the service of TCP (the case of SCTP would be similar).

Server Process

The server process starts first. It calls the *socket* function to create a socket, which we call the *listen* socket. This socket is only used during connection establishment. The server process then calls the *bind* function to bind this connection to the socket address of the server computer. The server program then calls the *accept* function. This function is a blocking function; when it is called, it is blocked until the TCP receives a connection request (SYN segment) from a client. The *accept* function then is unblocked and creates a new socket called the connect socket that includes the socket address of the client that sent the SYN segment. After the *accept* function is unblocked, the server knows that a client needs its service. To provide concurrency, the server process (parent process) calls the *fork* function. This function creates a new process (child process), which is exactly the same as the parent process. After calling the *fork* function, the two processes are running concurrently, but each can do different things. Each process now has two sockets: listen and connect sockets. The parent process entrusts the duty of serving the client to the hand of the child process and calls the accept function again to wait for another client to request connection. The child process is now ready to serve the client. It first closes the listen socket and calls the *recv* function to receive data from the client. The *recv* function, like the *recvfrom* function, is a blocking



function; it is blocked until a segment arrives. The child process uses a loop and calls the *recv* function repeatedly until it receives all segments sent by the client. The child process then gives the whole data to a function (we call it *handleRequest*), to handle the request and return the result. The result is then sent to the client in one single call to the *send* function. We need to emphasize several points here. First, the flow diagram we are using is the simplest possible one. The server may use many other functions to receive and send data, choosing the one which is appropriate for a particular application. Second, we assume that size of data to be sent to the client is so small that can be sent in one single call to the *send* function; otherwise, we need a loop to repeatedly call the *send* function. Third, although the server may send data using one single call to the *send* function, TC may use several segments to send the data. The client process, therefore, may not receive data in one single segment, as we will see when we explain the client process. Part a in the figure shows the status before the *accept* function returns. The parent process uses the *listen* socket to wait for request from the clients. When the accept function is blocked and returned (part b), the parent process has two sockets: the *listen* and the *connect* sockets. The client is connected to the *connect* socket. After calling the *fork* function (part c), we have two processes, each with two sockets. The client is connected to both processes.The parent needs to close its *connect* socket to free itself from the client and be free to listen to requests from other clients (part d). Before the child can start serving the connected client, it needs to close its *listen* socket so that a future request does not affect it (part e). Finally, when the child finishes serving the connected client, it needs to close its connect socket to disassociate itself from the client that has been served (part f). is the simplest possible one. The server may use many other functions to receive and send data, choosing the one which is appropriate for a particular application. Second, we assume that size of data to be sent to the client is so small that can be sent in one single call to the *send* function; otherwise, we need a loop to repeatedly call the *send* function. Third, although the server may send data using one single call to the *send* function, TCP may use several segments to send the data. The client process, therefore, may not receive data in one single segment, as we will see when we explain the client process. Part a in the figure shows the status before the *accept* function returns. The parent process uses the *listen* socket to wait for request from the clients. When the accept function is blocked and returned (part b), the parent process has two sockets the *listen* and the *connect* sockets. The client is connected to the *connect* socket. After calling the *fork* function (part c), we have two processes, each with two sockets. The client is connected to both processes.The parent needs to close its *connect* socket t free itself from the client and be free to listen to requests from other clients (part d). Before the child can start serving the connected client, it needs to close its *liste* socket so that a future request does not affect it (part e). Finally, when the child finishes serving the connected client, it needs to close its connect socket to disassociate itself from the client that has been served (part f).