BITS
NARSAMPET

Department of CSE

Laboratory Manual	
Course:	B.Tech.
Year & Semester:	III – I
Class:	CSE
Subject:	**Computer Networks Lab Manual**
Regulation:	R22

BALAJI INSTITUTE OF TECHNOLOGY AND SCIENCE (AUTONOMOUS)

B.Tech (Department of Computer Science & Engineering)

COMPUTER NETWORKS LAB

Course Outcomes:

- Implement data link layer farming methods.
- Analyze error detection and error correction codes.
- Implement and analyze routing and congestion issues in network design.
- Implement Encoding and Decoding techniques used in presentation layer.
- To be able to work with different network tools.

List of Experiments:

1. Implement the data link layer framing methods such as character, character-stuffing and bitstuffing.
2. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP.
3. Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.
4. Implement Dijkstra's algorithm to compute the shortest path through a network
5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.
6. Implement distance vector routing algorithm for obtaining routing tables at each node.
7. Implement data encryption and data decryption.
8. Write a program for congestion control using Leaky bucket algorithm.
9. Write a program for frame sorting techniques used in buffers.
10. Write a program for congestion control using Token bucket algorithm.

Index

1. COMPUTER NETWORKS

Definition:

A computer network is a group of computer systems and computing hardware devices that are linked to gether through communication channels to facilitate communication and resource-sharing among a wide range of users.

Applications:

Major applications of computer networks are:

1. Business Applications
2. Home Applications
3. Mobile Computers

1. **Business Applications:**

- Use of computers in business field to monitor production, inventories, to make payments.
- Resource sharing is the important purpose of using computer networks.

2. Home Applications:

Use of computers in home is widespread. Popular uses of computers in home are:

- Internet access.
- Personal Communication.
- Entertainment.
- Electronic Commerce.

3. **Mobile Computers:**

Many users use desktop computers at office and want to be connected to the office network while traveling. This is possible by wireless networks. Wireless networks are used in:

- Taxis, delivery vehicles and other mobile vehicles for keeping contacts with their office.
- Geographical Information Systems (GIS).
- Military applications.

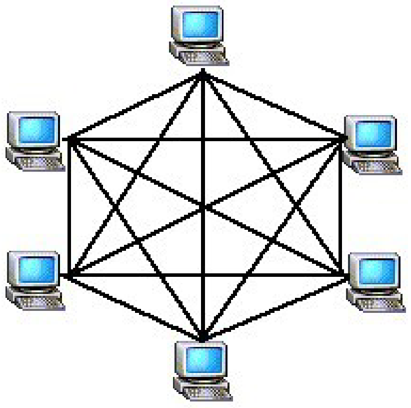
- Air ports.
- Banking
- Weather reporting.

2. NETWORK TOPOLOGIES

- A network topology is the geometric representation of the all the links and linking devices to one another
- Five types of topologies are commonly used in the network. They are bus, star, ring, mesh and hybrid topology.

Mesh Topology:

- In a mesh network topology, each of the network nodes are interconnected with one another.
- Every node not only sends its own signals but also relays data from other nodes. It is commonly used in wireless networks.
- Flooding or routing technique is used in mesh topology.



Mesh Topology Diagram

Advantages of Mesh topology:

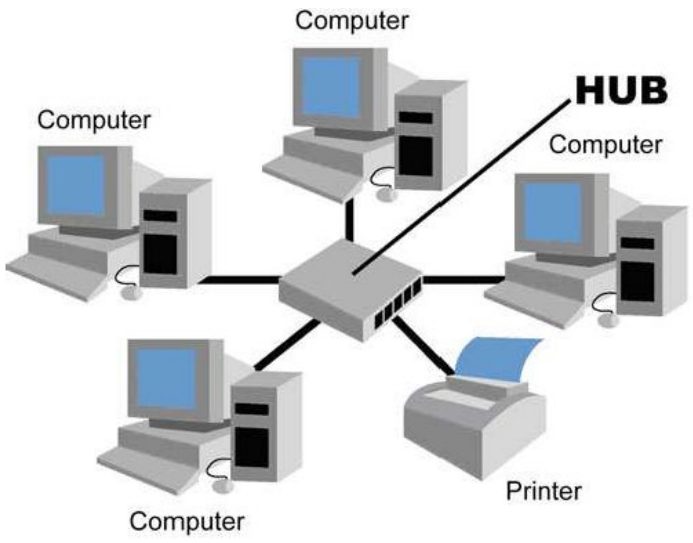
1. Troubleshooting is easy.
2. Isolation of network failures is easy.

Disadvantages of Mesh topology:

1. Difficulty of installation.
2. Costly because of maintaining redundant links.
3. Difficulty of reconfiguration.

Star Topology:

- In Star topology, all the components of network are connected to the central device called "hub".
- In this topology all the workstations are connected to central device with a point-to-point connection.



Star Topology Diagram

Advantages of Star Topology:

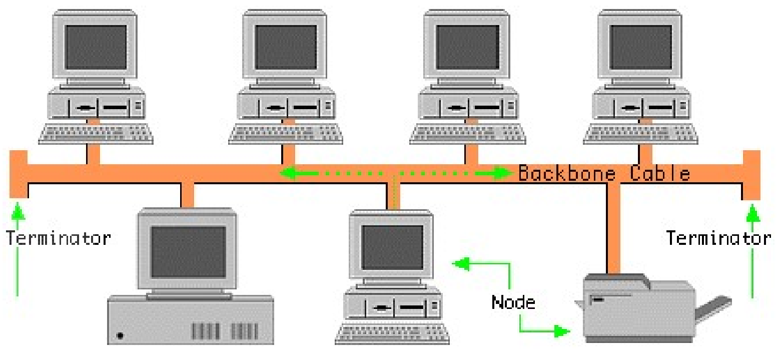
1. In star topology new nodes can be added or removed easily without affecting rest of the network
2. Troubleshooting is easy.

Disadvantages of Star Topology:

1. If the central hub fails, the whole network fails to operate.
2. Each device requires its own cable segment.

Bus Topology:

- Bus Topology is the simplest of network topologies. In this all the nodes are connected to the single cable, by the help of interface connectors.
- This central cable is the backbone of the network.



Bus topology diagram

Advantages Bus Topology:

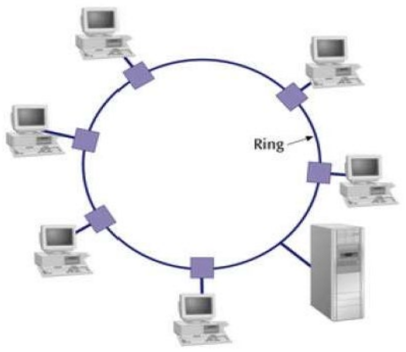
1. Easy to use and easy to install.
2. Bus topology costs very less.
3. Linear Bus network is mostly used in small networks.

Disadvantages Bus Topology:

1. It is difficult to detect and troubleshoot fault at individual station.
2. Maintenance costs can go higher with time.
3. Efficiency of Bus network reduces as the number of devices increases.

Ring Topology:

- In Ring Topology, all the nodes are connected to each-other in suc   a way that they make a closed loop.
- Data travels around the network, in one direction.



Advantages of Ring Topology:

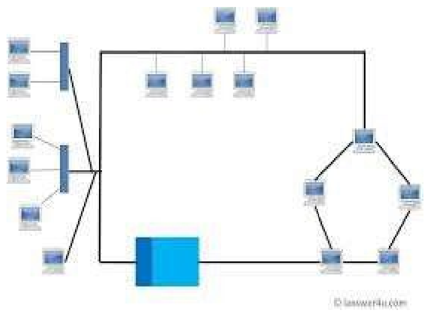
1. Cable failures are easily found.
2. Additional components do not affect the performance of network.

Disadvantages of Ring Topology:

1. It is difficult to troubleshoot a ring network.
2. Cost of cable is more in ring network.

Hybrid Topology:

- A network can be hybrid. Consider a main star topology with each branch connecting several stations in a bus topology.



Advantages of Hybrid Topology:

1. Unlike other networks, fault detection and troubleshooting is easy in this type of topology.
2. It is the combination of two or more topologies.

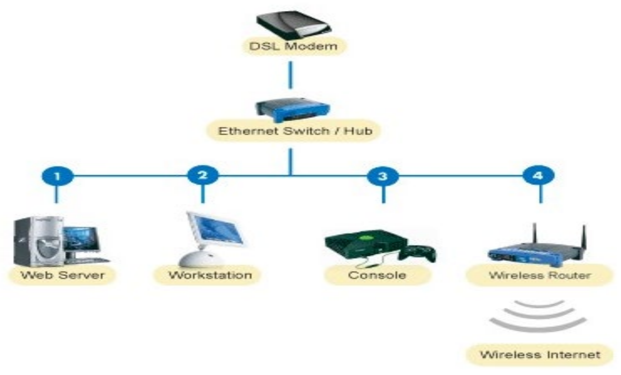
Disadvantages of Hybrid Topology:

1. One of the biggest drawbacks of hybrid topology is its design.
2. The hubs used to connect two distinct networks, are very expensive.

3. NETWORK TYPES

LAN - Local Area Network:

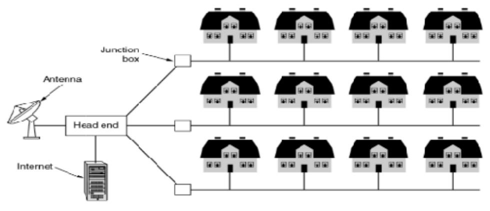
- A LAN connects network devices over a relatively short distance.
- The LAN transmission capacity is more than 1Mbps.
- The geographical coverage of LAN's is limited to areas less than 5 square kilometers.
- A networked office building, school, or home usually contains a single LAN.



MAN-Metropolitan Area Network:

- **A MAN** (*Metropolitan Area Networks*) connects multiple geographically nearby LANs to one another over an area of up to a few kilometers at high speeds.
- Thus, a MAN lets two remote nodes communicate as if they were part of the same local area network.
- A MAN is made from switches or routers connected to one another with high-speed links usually fiber optic cables.

Metropolitan Area Networks

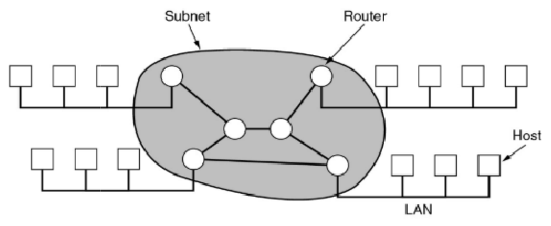


A metropolitan area network based on cable TV.

WAN-Wide Area Network:

A **WAN** spans a large geographic area, such as a state, province or country. WANs often connect multiple smaller networks, such as local area networks (LAN) or metro area networks (MAN). The world's most popular WAN is the Internet.

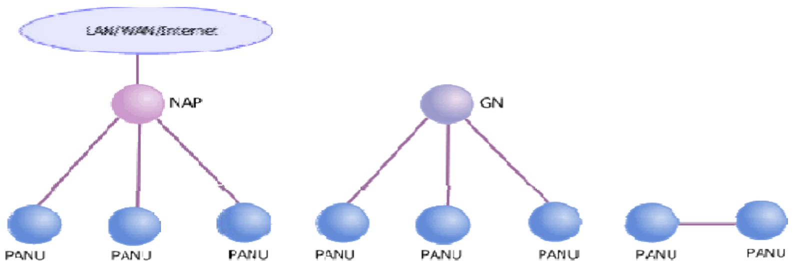
Wide Area Networks



Relation between hosts on LANs and the subnet.

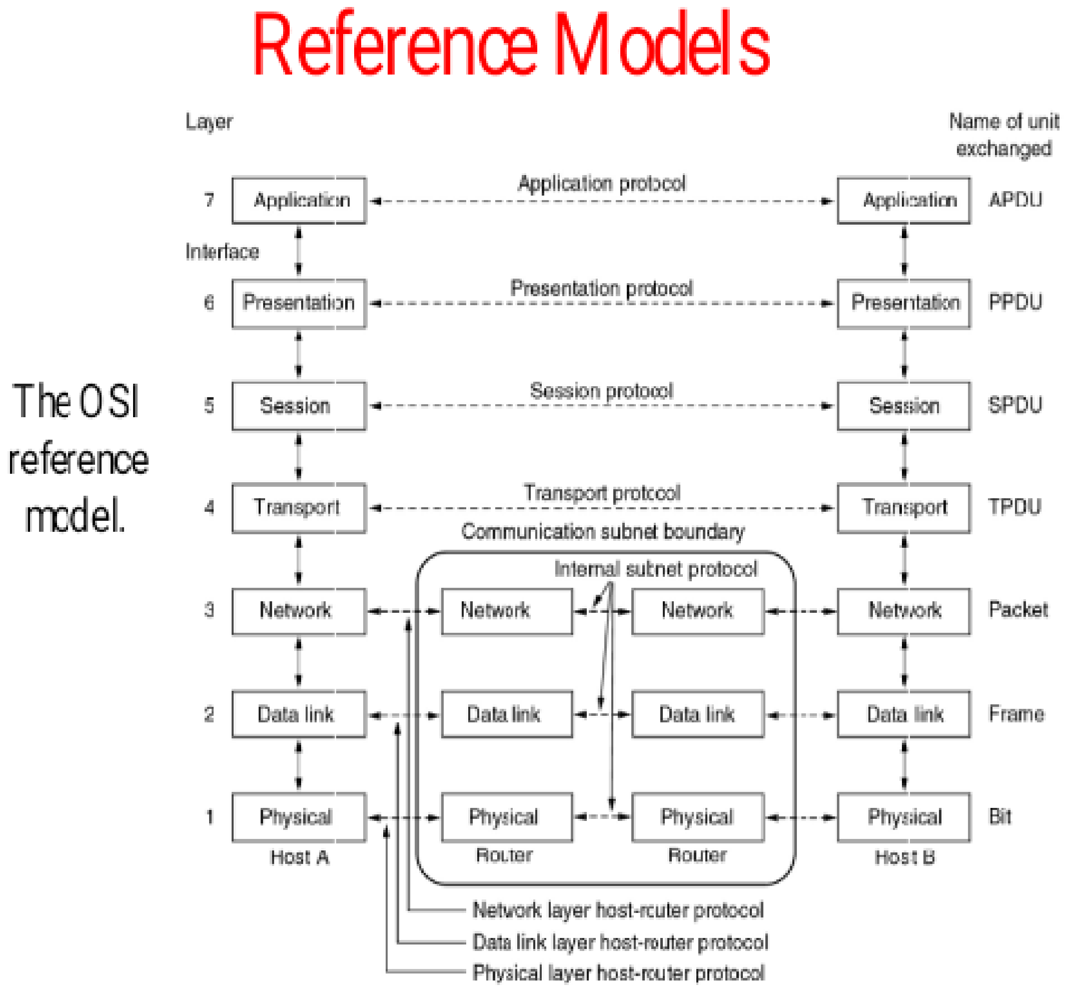
PAN-Personal Area Network:

A personal area network (PAN) is the interconnection of information technology devices within the range of an individual person, typically within a range of 10 meters.

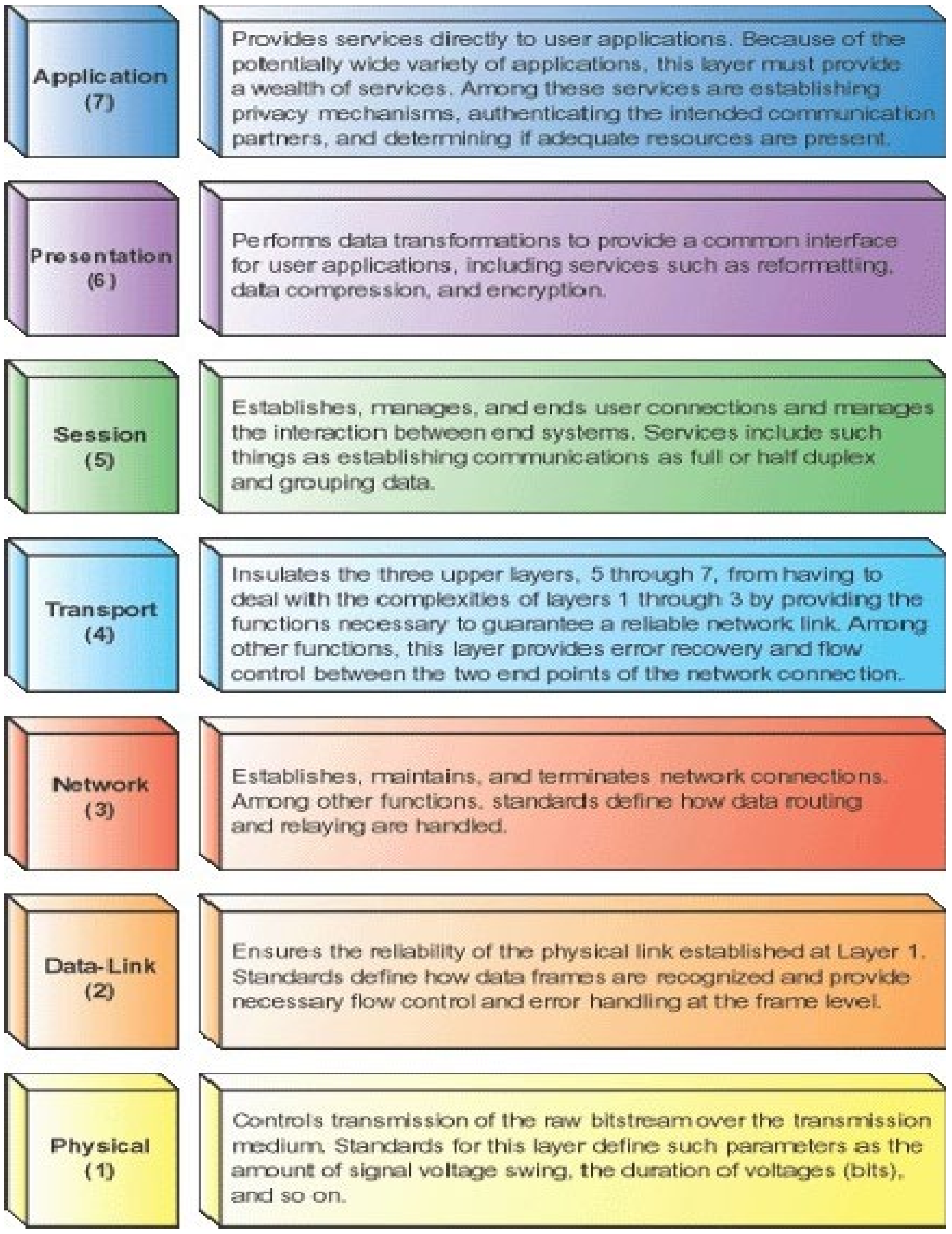


4. OSI / ISO MODEL

- The International Standards Organization for Standardization (ISO) developed the Open System Interconnection (OSI) reference model. OSI model is most widely used model for networking.
- An Open System is a set of protocols that allows any two different systems to communicate.



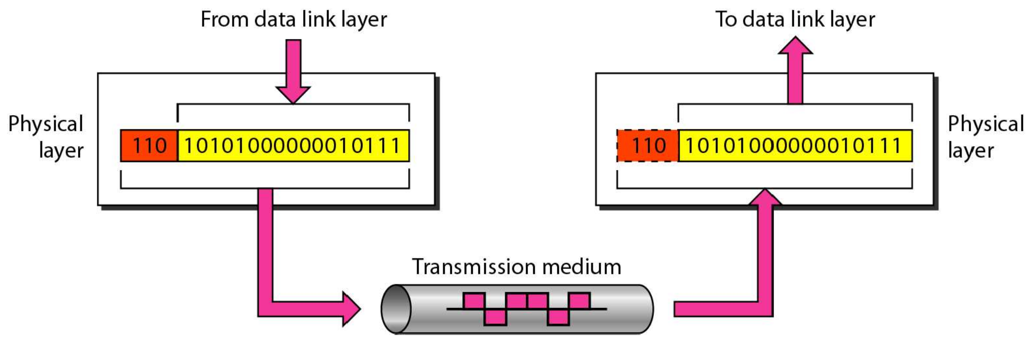
Seven layers of OSI model:



Application (7)	Provides services directly to user applications. Because of the potentially wide variety of applications, this layer must provide a wealth of services. Among these services are establishing privacy mechanisms, authenticating the intended communication partners, and determining if adequate resources are present.
Presentation (6)	Performs data transformations to provide a common interface for user applications, including services such as reformatting, data compression, and encryption.
Session (5)	Establishes, manages, and ends user connections and manages the interaction between end systems. Services include such things as establishing communications as full or half duplex and grouping data.
Transport (4)	Insulates the three upper layers, 5 through 7, from having to deal with the complexities of layers 1 through 3 by providing the functions necessary to guarantee a reliable network link. Among other functions, this layer provides error recovery and flow control between the two end points of the network connection.
Network (3)	Establishes, maintains, and terminates network connections. Among other functions, standards define how data routing and relaying are handled.
Data-Link (2)	Ensures the reliability of the physical link established at Layer 1. Standards define how data frames are recognized and provide necessary flow control and error handling at the frame level.
Physical (1)	Controls transmission of the raw bitstream over the transmission medium. Standards for this layer define such parameters as the amount of signal voltage swing, the duration of voltages (bits), and so on.

PHYSICAL LAYER:

- Physical layer is the lowest layer of the OSI model.
- The Physical Layer defines the electrical and physical specifications for devices.



Functions of Physical layer:

1. **Physical characteristics of interfaces and media:**
   - The physical layer defines the characteristics of the interface between the devices and the transmission medium.
   - It also defines the type of transmission medium.

2. **Representation of bits:**
   - The physical layer data consists of a stream of bits (sequence of 0's or 1's) with no interpretation.
   - To be transmitted, bits must be encoded into signals electrical or optical. The physical layer defines the type of encoding.

3. **Data rate:**
   The transmission rate the number of bits sent each second is also defined by the physical layer.

4. **Synchronization of bits:**
   The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level.

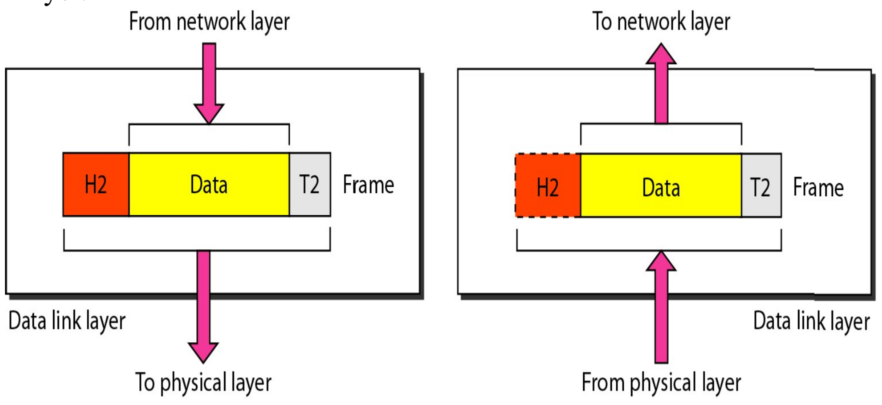
5. **Line configuration:**
   - The Physical layer is concerned with the connection devices to the media.
   - In a point-to-point configuration, two devices are connected through a dedicated link. In a multipoint configuration, a link is shared among several devices.

6. **Physical topology:**
   - The physical topology defines how devices are connected to make a network.
   - Devices can be connected by using a mesh topology, a star topology, a ring topology, or a hybrid topology.

7. **Transmission mode:**
   - The physical layer also defines the direction of transmission between two devices: simplex, half-duplex. Full-duplex.
   - In simplex mode, only one device can send, the other receives. The simplex mode is a one-way communication.
   - In the half-duplex mode, two devices can send and receive, but not at same time.
   - In full-duplex mode, two devices can send and receive at the same time.

DATA LINK LAYER:

- The data link layer is responsible for transmitting the frames from one node to the next.
- It transforms the physical layer to a reliable link making it an error free link to upper layer.



Functions of Data Link Layer:

1. **Framing:**
   - The frames received from network layer are divided into manageable data units called frames.

2. **Physical addressing:**
   - When 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 receiver of the frame.
   - If the frame is intended for a system outside the sender's to the next one.

3. **Flow control:**
   - If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.

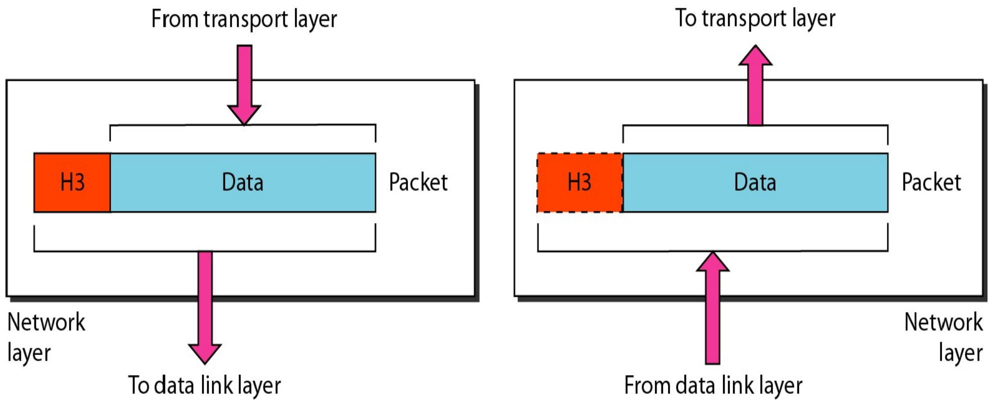
4. **Error control:**
   - The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplication frames.
   - Error control is normally achieved through a trailer added to the end of the frame.

5. **Access control:**

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

NETWORK LAYER:

The network layer is responsible for the delivery of packets from the source to destination across multiple networks.

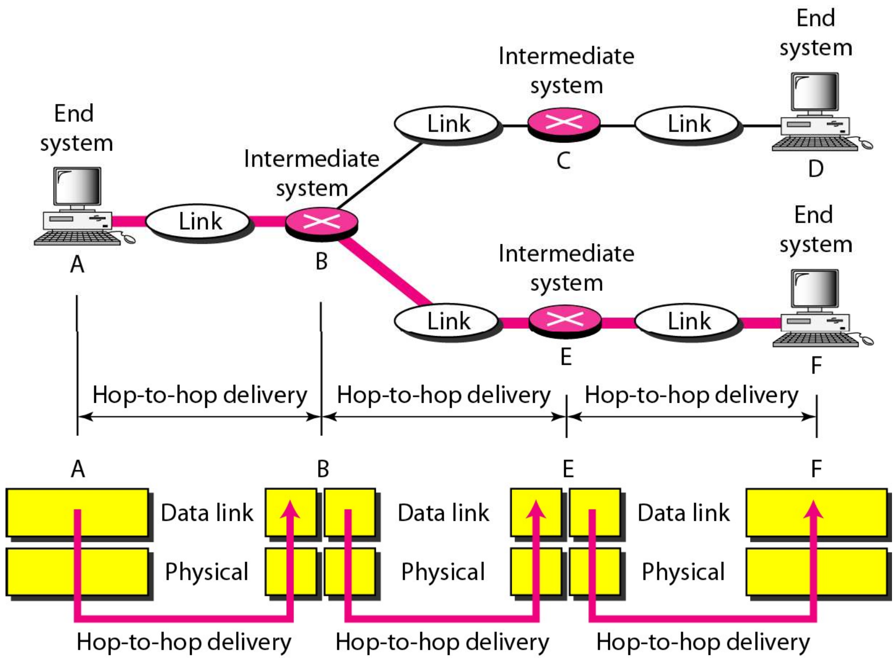


Functions of Network layer:

1. **Logical addressing:**
   - The physical addressing implemented by the data link layer handles th   addressing problem locally.

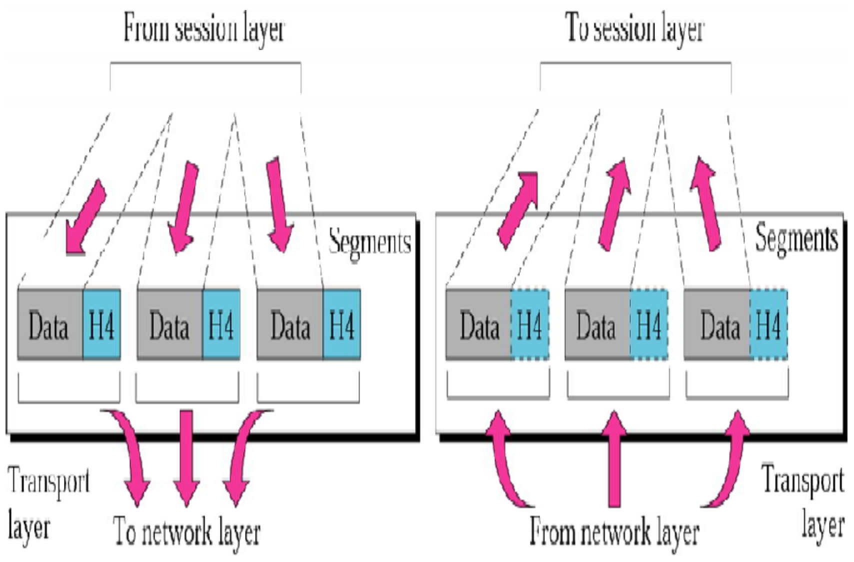
   - If a packet passes the network boundary, we need another addressing system to help distinguish the source and destination systems.

2. **Routing:**
   - When independent networks or links are connected to create internet works 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



TRANSPORT LAYER:

The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on a host.



Functions of transport layer:

1. **Service-point addressing:**
   - Computer performs several operations simultaneously. Process-to-process delivery means specific process of one computer must be delivered to specific process on other computer. The transport layer header therefore includes port address.
   - Network layer delivers packet to the desired computer and transport layer, gets message to the correct process on that computer.

2. **Segmentation and reassembly:**

- A message is divided into segments, each segment contains a sequence number which enables transport layer to reassemble at destination.

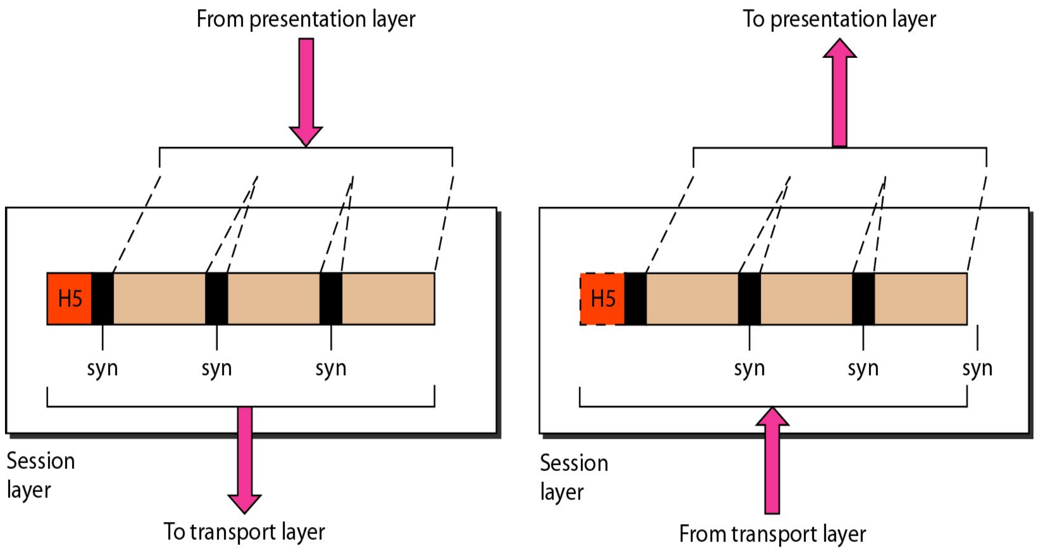
3. **Connection control:**
   - Transport layer performs connectionless or connection oriented services with the destination machine.

4. **Flow control:**
   - Transport layer performs end-to-end flow control while data link layer performs it across the link.

5. **Error control:**
   - Error control at this layer is performed on end-to-end basis rather than across the link. The transport layer ensures error free transmission.

SESSION LAYER:

The session layer is network dialog controller that is it establishes and synchronizes the interaction between communication systems.



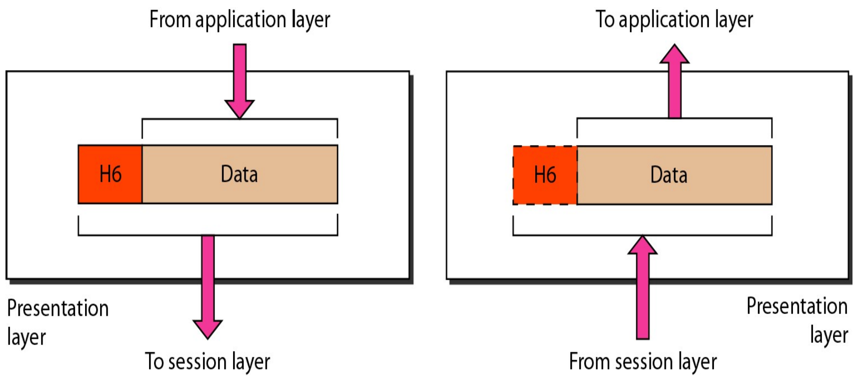
Functions of session layer:

1. **Dialog control:**
   - Communication between two processes takes place in either half duplex or full duplex mode. The session layer manages dialog control for this communication.

2. **Synchronization :**
   - Session layer adds synchronization points into stream of data.

PRESENTATION LAYER:

The presentation layer deals with syntax and semantics of the information being exchanged.



Functions of Presentation Layer:

1. Translation:

- Different computers use different encoding systems. The presentation layer maintains interoperability between the two encoding systems.

2. Encryption:

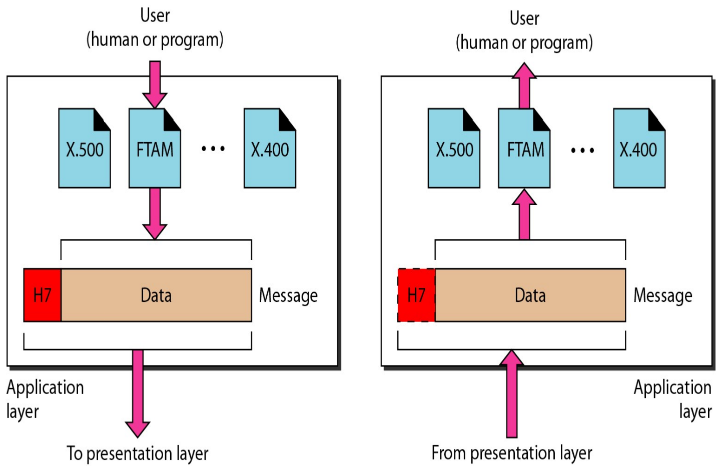
- Encryption is transforming sender information to other form to ensure privacy while transmission. Decryption is a reverse process.

3. **Compression:**

- Compression is a technique of reducing number of bits required to represent the data.

APPLICATION LAYER:

Application layer is responsible for accessing the network by user. It provides user interfaces and other supporting services such as e-mail, remote, file access, file transfer, sharing database, message handling, and directory services.



Functions of Application Layer:

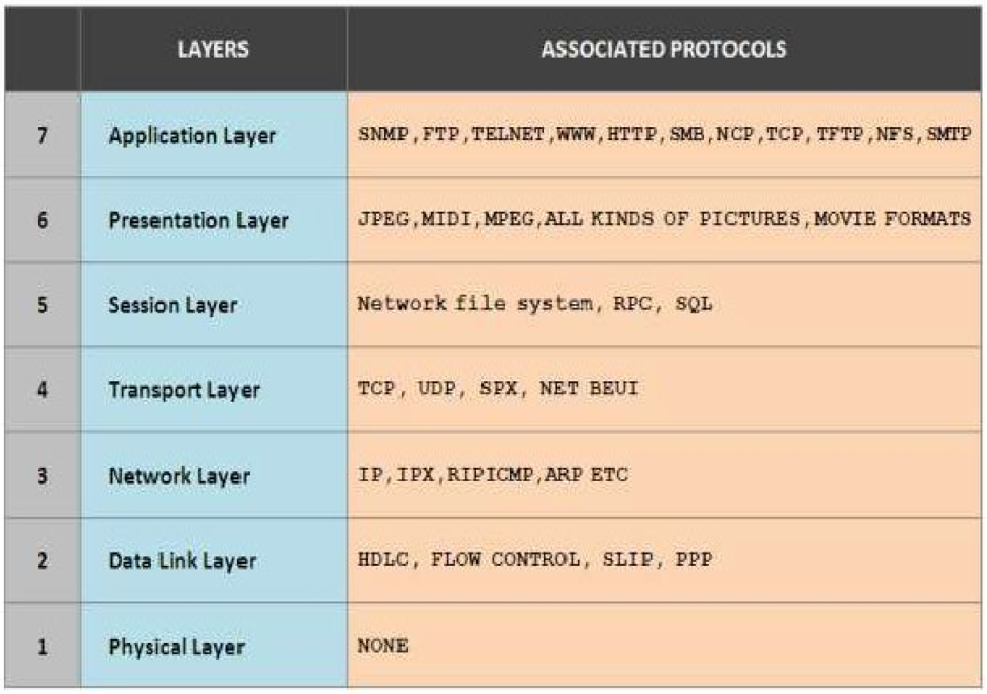
1. **Network Virtual Terminal:**
   - It is a software version of physical terminal that allows a user to log onto a remote host.

2. **File Transfer, Access and Management (FTAM):**
   - FTAM allows user to access files in remote hosts, to retrieve files and to manage files in remote computer.

3. **Mail Services:**
   - E-mail forwarding, storage are the services under this category.

4. **Directory Services:**
   - Directory services include access for global information and distributed database.

PROTOCOLS IN OSI LAYERS:

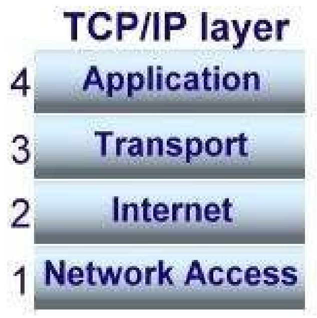


	LAYERS	ASSOCIATED PROTOCOLS
7	Application Layer	SNMP,FTP,TELNET,WWW,HTTP,SMB,NCP,TCP,TFTP,NFS,SMTP
6	Presentation Layer	JPEG,MIDI,MPEG,ALL KINDS OF PICTURES,MOVIE FORMATS
5	Session Layer	Network file system, RPC, SQL
4	Transport Layer	TCP, UDP, SPX, NET BEUI
3	Network Layer	IP,IPX,RIPICMP,ARP ETC
2	Data Link Layer	HDLC, FLOW CONTROL, SLIP, PPP
1	Physical Layer	NONE

5. TCP/IP PROTOCOL

- TCP/IP stands for transmission protocol/Internet protocol.

- The TCP/IP reference model is a set of protocols that allow communication across multiple diverse networks.
- TCP/IP is normally considered to be a four layer system. Layers of TCP/IP are Application layer, Transport layer, Internet layer, Host to network layer.
- Host to network layer is also called physical and data link
- The application layer in TCP/IP can be equated with the combination of session, presentation, application layer of the OSI reference model.



- TCP/IP defines two protocols at transport layer: TCP and UDP.
- **User Datagram Protocol (UDP)** is connectionless protocol.
- UDP is used for application that requires quick but necessarily reliable delivery.
- Internet layer also called network layer. Internet layer handles communication from one machine to the other. Routing of packet takes place in internet layer.
- TCP/IP does not define any specific protocol in host to network layer. This layer is responsible for accepting and transmitting IP data grams.
- This layer normally includes the device driver in the operating system.
- Detail function of each layer is given below.

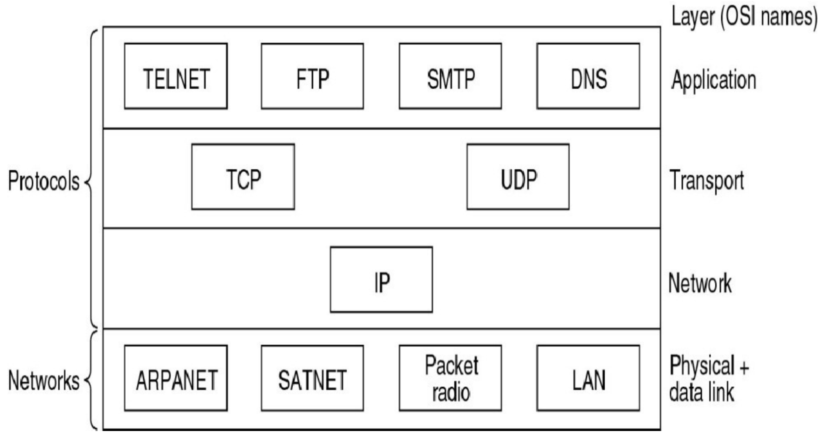
1. Application layer:

- Application layer includes all process and services that use the transport layer to deliver data. The most widely known application protocols are: TELNET, File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Simple Network Management Protocol (SNMP).
- TELNET is the Network Terminal Protocol, which provides remote login over the network.
- FTP is used for interactive file transfer.
- SMTP delivers the electronic mail.

2. Transport layer:

- Application programs send data to the transport layer protocols TCP and UDP.
- An application is designed to choose either TCP or UDP based on the services it needs.
- The transport layer provides per entities on the source and destination hosts to carry on a conversation. Both end’s protocol is defined in this layer.
- TCP is reliable connection oriented protocol that allows a byte stream originating on one computer to be delivered without error to any other computer in the internet.
- It converts the incoming byte stream into discrete message and passes each one onto the Internet layer.
- At the destination side, the receiving TCP reassembles the received data or messages into the output format.

- TCP also handles flow control. It synchronizes between fast sender and slow receiver.
- UDP is a connectionless protocol. Sometimes this type of protocol is used for prompt delivery.



3. **Internet layer :**
   - The Internet network level protocol (IP, ARP, ICMP) handle machine to machine communications.
   - These protocols provide for transmission and reception of transport requests and handle network level control.

   - The TCP/IP Internet layer moves data from one host to another even if the hosts are on different networks.
   - The primary protocol used to move data is the Internet Protocol (IP) which provides the following services:

     a. Addressing:
     - Determining the route to deliver data to the destination **host.**

     b. Fragmentation:
     - Breaking the messages into pieces if an intervening network cannot handle a large message.
   - It provides a connectionl ss method of delivering data from one host to another. It does not guarantee delivery and does not provide sequencing of data grams.
   - It attaches a header to datagram that includes source address and the destination address, both of which are unique internet addresses.

4. **Host to network layer:**

   - This layer is also called network interface layer.
   - This layer is same as physical and data link layer of OSI model. Host to network layer cannot define any protocol.
   - It is responsible for accepting and transmitting IP data grams.
   - This layer may consist of a device driver in the operating system and the corresponding network interface card in the machine.

- **HYPER TEXT TRANSFER PROTOCOL (HTTP):**

  Web servers implement this protocol. Short for Hyper Text Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

- **SIMPLE MAIL TRANSFER PROTOCOL (SMTP):**

Used by e-mail servers (and sometimes Web servers) to send e-mail. Short for Simple Mail Transfer Protocol, a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP. In addition, SMTP is generally used to send messages from a mail client to a mail server. This is why you need to specify both the POP or IMAP server and the SMTP server when you configure your e- mail application.

- **INTERNET MESSAGE ACCESS PROTOCOL (IMAP):**

Short for Internet Message Access Protocol, a protocol for retrieving e-mail messages. The latest version, IMAP4, is similar to POP3 but supports some additional features. For example, with IMAP4, you can search through your e-mail messages for keywords while the messages are still on mail server. You can then choose which messages to download to your machine.

- **FILE TRANSFER PROTOCOL (FTP):**

The protocol for exchanging files over the Internet. FTP works in the same way as HTTP for transferring Web pages from a server to a user's browser and SMTP for transferring electronic mail across the Internet in that, like these technologies, FTP uses the Internet's TCP/IP protocols to enable data transfer.

FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server (e.g., uploading a Web page file to a server).

- **DOMAIN NAME SYSTEM (DNS):**

Short for Domain Name System (or Service or Server), an Internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.example.com might translate to 198.105.232.4.

- **TRANSMISSION CONTROL PROTOCOL (TCP):**

Creates a reliable connection between two computers. TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.

- **INTERNET PROTOCOL (IP):**

Provides addressing scheme. IP specifies the format of packets, also called datagrams, and the addressing scheme. Most networks combine IP with a higher-level protocol called

Transmission Control Protocol (TCP), which establishes a virtual connection between a destination and a source.

IP by itself is something like the postal system. It allows you to address a package and drop it in the system, but there's no direct link between you and the recipient. The current version of IP is IPv4. A new version, called IPv6 is under development.

- **INTERNET CONTROL MESSAGE PROTOCOL (ICMP):**

  Provides error messages. An extension to the Internet Protocol (IP) defined by RFC 792. ICMP supports packets containing error, control, and informational messages. The PING

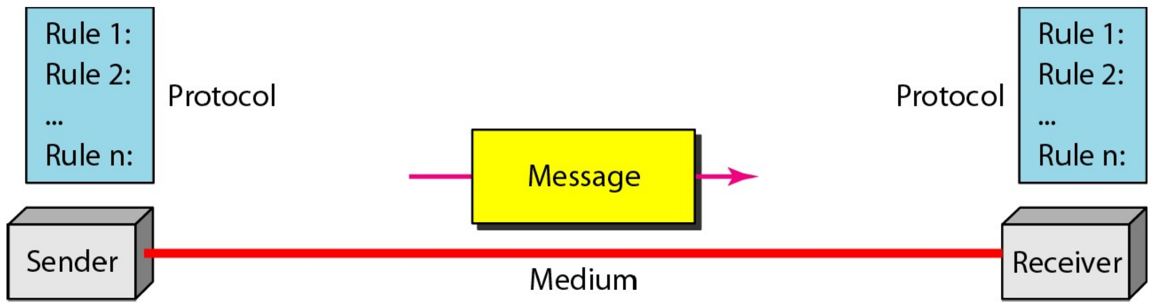
  command, for example, uses ICMP to test an Internet connection.

- **USER DATAGRAM PROTOCOL (UDP):**

  A connectionless protocol that, like TCP, runs on top of IP networks. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. It's used primarily for broadcasting messages over a network

6. Explain about elements of Data communication systems?

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



* Protocol: A protocol is a set of rules that govern data communications.it represents an agreement between communication devices.without a proto ol ,two devices may be connected but not communicate.

7. List and brief about Guided and unguided media in a tabular form?

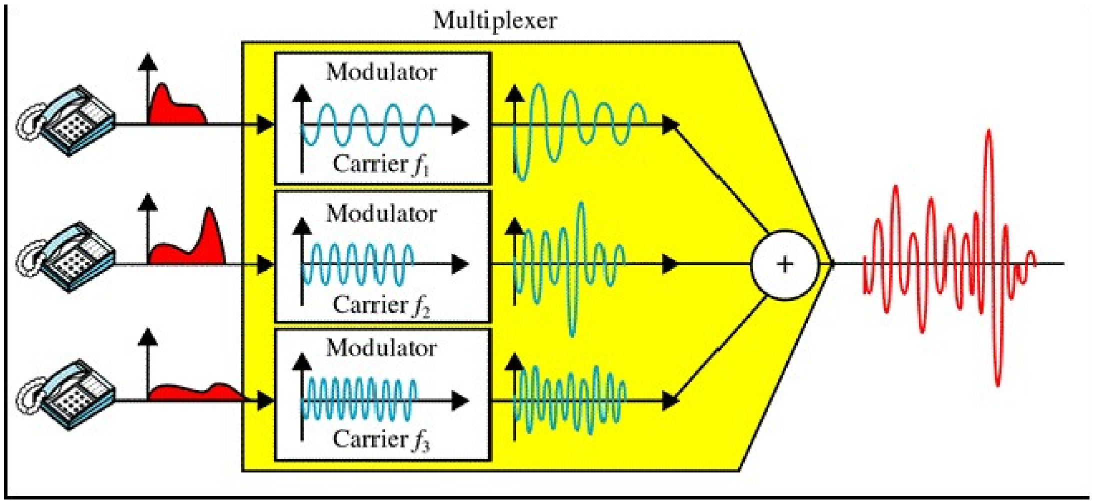
GUIDED MEDIA	UNGUDED MEDIA

➢ Guided media, which are those that provide a conduit from one device to another	➢ Unguided media transport el ctromagnetic waves without using a physical conductor.
➢ This type of communication is often referred to as wired communication.	➢ This type of communication is often referred to as wireless communication.
➢ The signal energy propagates within the guided media .i.e. through wires.	➢ The signal energy propagates through air.
➢ It is mainly suited for point to point line configurations	➢ It is mainly used for broadcasting purpose.
➢ The signal propagates in the form of voltage, current or photons.	➢ The signal propagates in the form of electromagnetic waves.
➢ Examples are:- =>Twisted Pair Cable =>Co-axial Cable =>Optical Fiber Cable	➢ Examples are:- Wireless transmission waves =>radio waves =>Microwave =>Infrared
➢ Propagation mode: => Multi mode --step index --graded index =>single mode	➢ Propagation mode: =>ground propagation (below 2 MHz) =>sky propagation (2-30 MHz) =>line-of-sight propagation (above 30 MHz)
➢ Attenuation depends exponentially on the distance.	➢ Attenuation is proportional t square of distance.
➢ In this media medium is more important in setting transmission parameters.	➢ In this media bandwidth of the signal produced by transmitting antenna is important in setting transmission parameters.
➢ Coaxial cable carries signals of higher frequency ranges than twisted-pair cable	➢ Higher frequency signals can be focused in a directional beam. Lower frequency signals are omnidirectional.

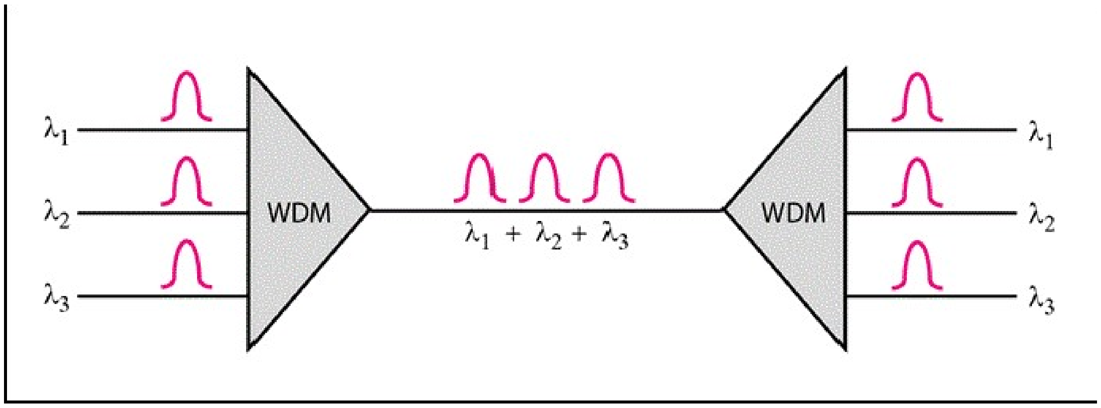
8. What are the different Types of multiplexing?

Multiplexing is of following three types:

1. **Frequency-division multiplexing** (FDM) is an analog multiplexing technique t at combines analog signals. FDM is applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted. In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth i.e **guard bands,** to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original data frequencies.

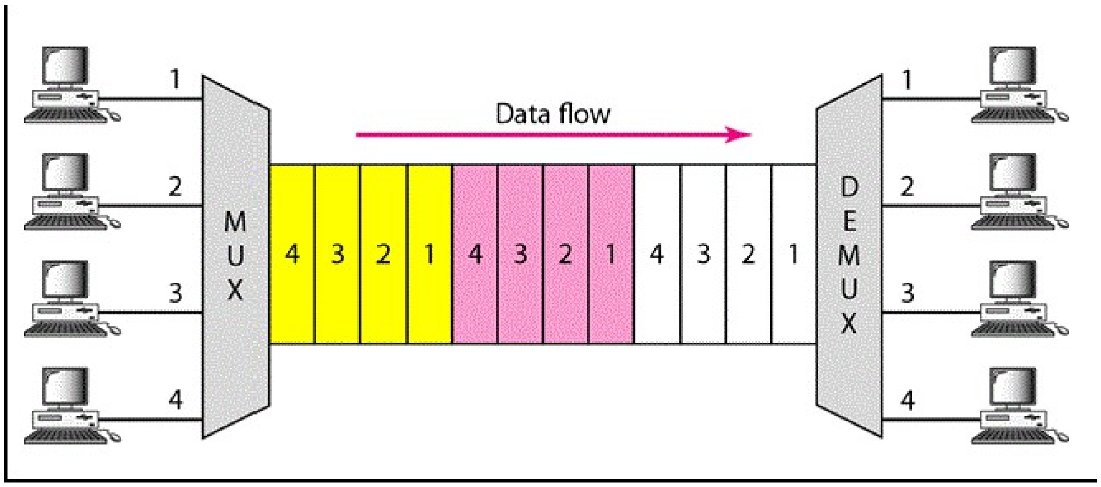


2. **Wavelength-division multiplexing** (WDM) is an analog multiplexing technique to combine optical signals.WDM is designed to use the high data rate capability of fiber-optic cable. The optical fiber data rate is higher than the data rate of metallic transmission cable. Using a fiber-optic cable for one single line wastes the available bandwidth. Multiplexing allows us to combine several lines into one. WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve optical signals transmitted through fiber-optic channels. The idea is the same: We are combining different signals of different frequencies. The difference is that the frequencies are very high.



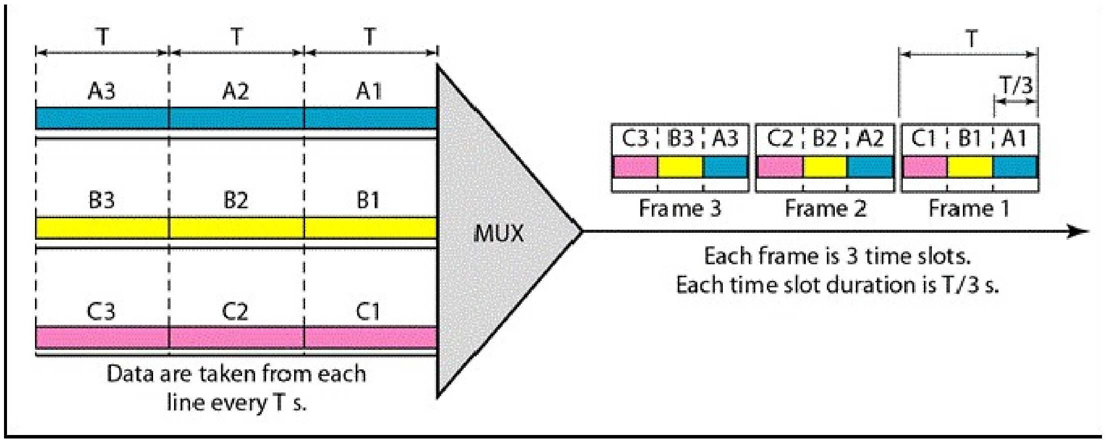
3. **(TDM)** is a digital multiplexing technique for combining several low-rate channels into one high-rate one.TDM is a digital process that allows several connections to share the high bandwidth of a link Instead of sharing a portion of the bandwidth as in FDM, time is shared. Each connection occupies a portion of time in the link.Digital data from different sources are combined into one

timeshared link. However, this does not mean that the sources cannot produce analog data; analog data can be sampled, changed to digital data, and then multiplexed by using TDM.



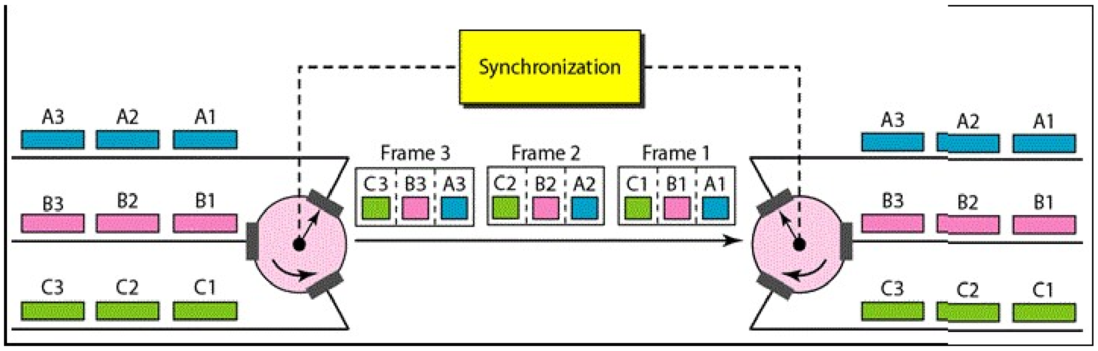
Types of TDM:

a. Synchronous TDM: In synchronous TDM, a round of data units from each input connection is collected into a frame (we will see the reason for this shortly). If we have n connections, a frame is divided into n time slots and one slot is allocated for each unit, one for each input line. If the duration of the input unit is T, the duration of each slot is Tin and the duration of each frame is T (unless a frame carries some other information, as we will see shortly). The data rate of the output link must be n times the data rate of a connection to guarantee the flow of data.Time slots are grouped into frames. A frame consists of one complete cycle of time slots, with one slot dedicated to each sending device. In a system with n input lines, each frame has n slots, with each slot allocated to carrying data from a specific input line.



T
T
T
A3
A2
A1
B3
B2
B1
C3
C2
C1
Data are taken from each line every T s.
MUX
T
T/3
C3 B3 A3
C2 B2 A2
C1 B1 A1
Frame 3
Frame 2
Frame 1
Each frame is 3 time slots.
Each time slot duration is T/3 s.

b. Interleaving:TDM can be visualized as two fast-rotating switches, one on the multiplexing side and the other on the demultiplexing side. The switches are synchronized and rotate at the same speed, but in opposite directions. On the multiplexing side, as the switch opens in front of a connection, that connection has the opportunity to send a unit onto the path. This process is called **interleaving**. On the demultiplexing side, as the switch opens in front of a connection, that connection has the opportunity to receive a unit from the path.



c. Statistical Time-Division Multiplexing: In synchronous TDM, each input has a reserved slot in the output frame. This can be inefficient if some input lines have no data to send. In statistical time-division multiplexing, slots are dynamically allocated to improve bandwidth efficiency. Only when an input line has a slot's worth of data to send is it given a slot in the output frame. In statistical multiplexing, the number of slots in each frame is less than the number of input lines. The multiplexer checks each input line in round robin fashion; it allocates a slot for an input line if the line has data to send; otherwise, it skips the line and checks the next line.

9. Explain about the following with neat diagrams

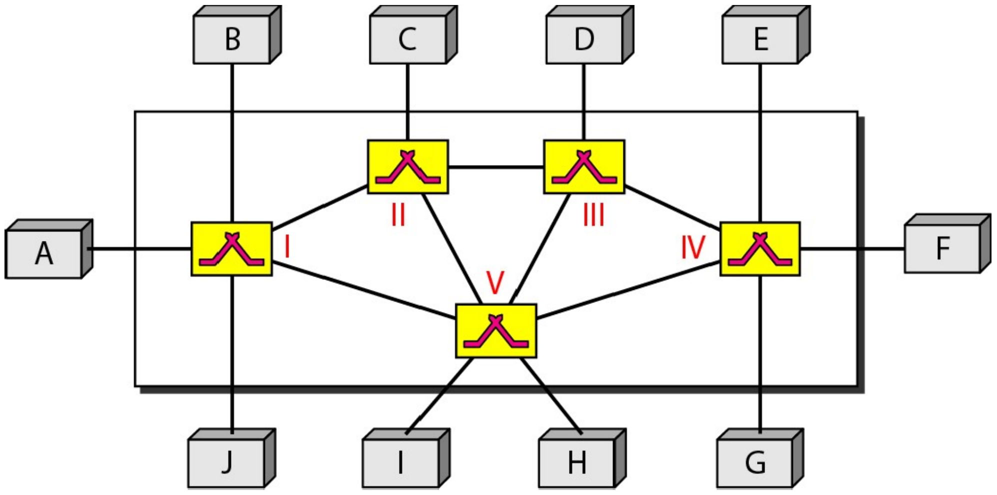
a. Circuit Switched networks
b. Virtual circuit networks
c. Datagram networks

Switched networks: A Switched network consists of a series of interlinked nodes called switches.

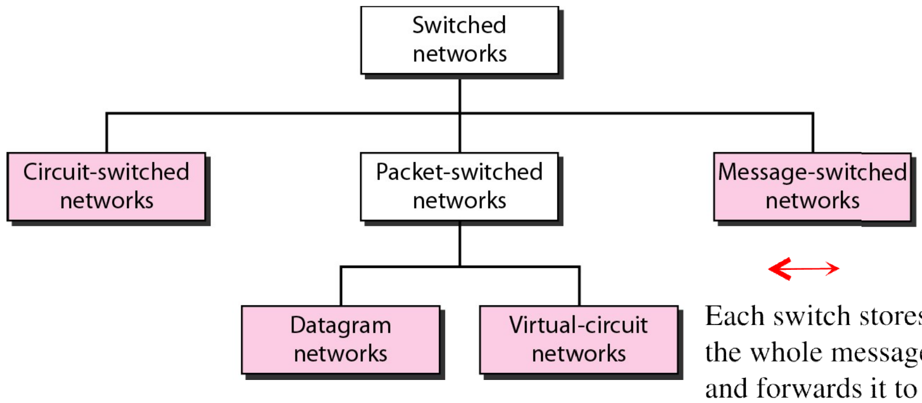
- A network is a set of connected devices. Whenever we have multiple stations/nodes, we have the problem of how to connect them to make one – to- one communication possible.
- To connect point-to-point every station (mesh) is impractical. Thousands of nodes with tons of millions of multi-links (?)
- In LANs this is achieved using one of three methods:
  - Via central controller (star)
  - Connection to common bus in a multipoint configuration (bus/hub)

Impossible also create a bus because the distance is too far.

- None of the previous works in larger networks with large physical separation or consisting of a large number of computers.
- **The solution is a switching / switched Network.**
- Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network some of these nodes are connected to the end systems. Others are used only for the routing.



Taxonomy of switched networks:



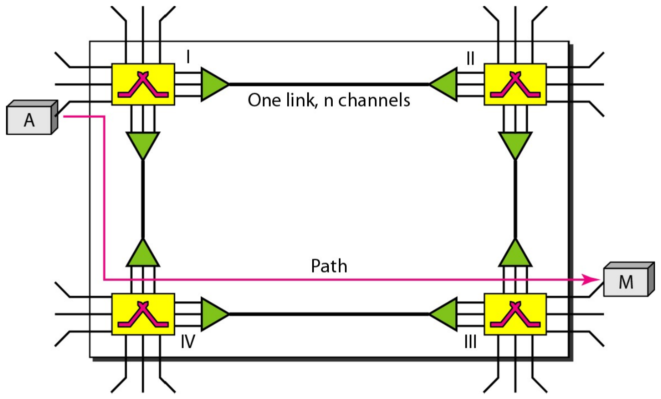
Each switch stores the whole message and forwards it to the next switch.

CIRCUIT-SWITCHED NETWORKS:

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links.

Each link is normally divided into n channels by using FDM or TDM.

Figure shows a trivial circuit- switched n/w with four switches and four links. Each link is divided into n (3) channels by using FDM or TDM.



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

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

Three Phases

The actual communication in a circuit-switched network requires three phases:

1. Connection setup
2. data transfer
3. Connection teardown.

Connection setup: means creating dedicated channels between the switches (system- system).

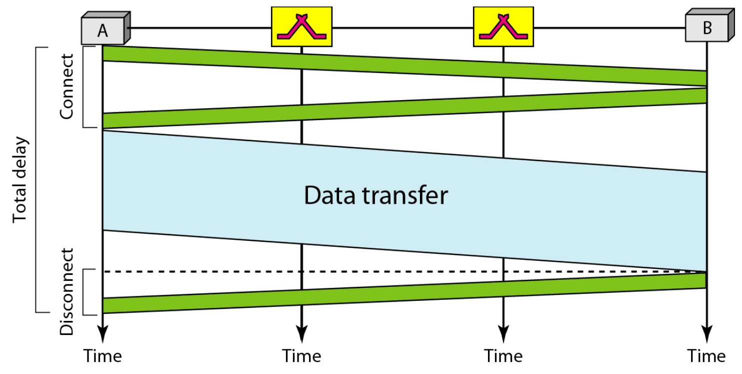
Data Transfer Phase: After the establishment of the dedicated circuit (channels), the two parties can transfer data.

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

Efficiency:

- Circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection.
- These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation.

Delay:



Delay=PTSCR+RSTT+PTAD+ASTT+DT+CTT

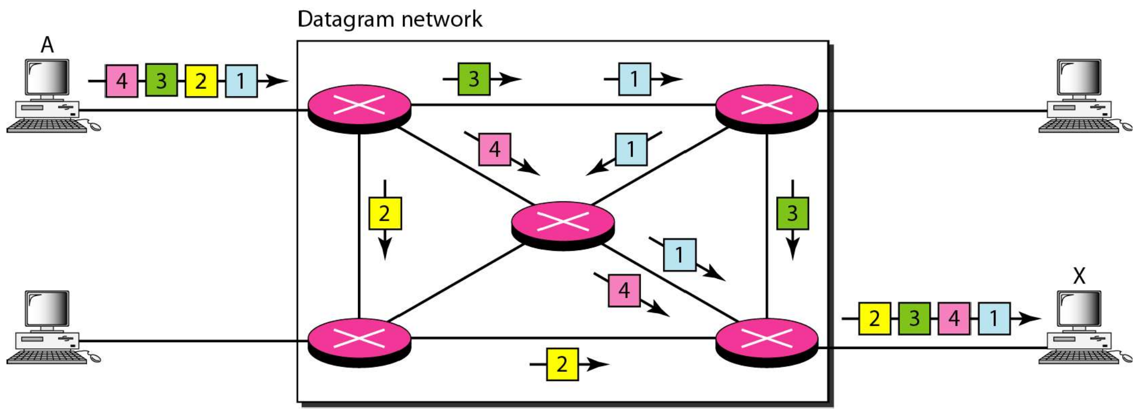
Switching at the physical layer in the traditional telephone network uses the circuit-switching approach.

DATAGRAM NETWORKS:

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

- Data are transmitted in discrete units called packets.
- Size of the packet depends on the protocol and network.
- Resource allocation is done on a FCFS basis.

- In datagram network, each packet is treated independently of all others.
- Switches here refer as router.
- Packets switched networks are connectionless, hence no resource allocation.
- Connectionless means the switch does not keep information about the connection state.
- There are no setup and teardown phases. Each packet is treated same by switch regardless of its source and destination.
- Datagram switching is done at network layer.
- Packets in this approach are referred to as Datagrams.

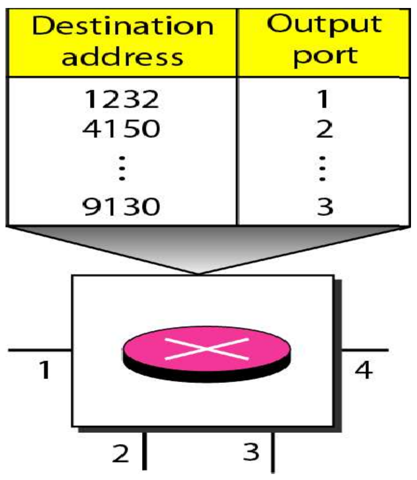


- ➢ In figure, all 4 packets belong to the same message, but may travel different path due to the lack of resources.
- ➢ Datagram travel out of order.
- ➢ May result packet dropped.
- ➢ Datagram approach can be referred as connectionless networks.

Routing table in a datagram network

- Addresses of sender and destination needed at each switch.
- Routing table is managed at each switch.
- Routing tables are dynamic and updated periodically.
- The destination addresses and the corresponding forwarding output ports are recorded in the tables. Refer fig 8.8
- A switch in a datagram network uses a routing table that is based on the destination address.

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



Destination address	Output port
1232	1
4150	2
⋮	⋮
9130	3

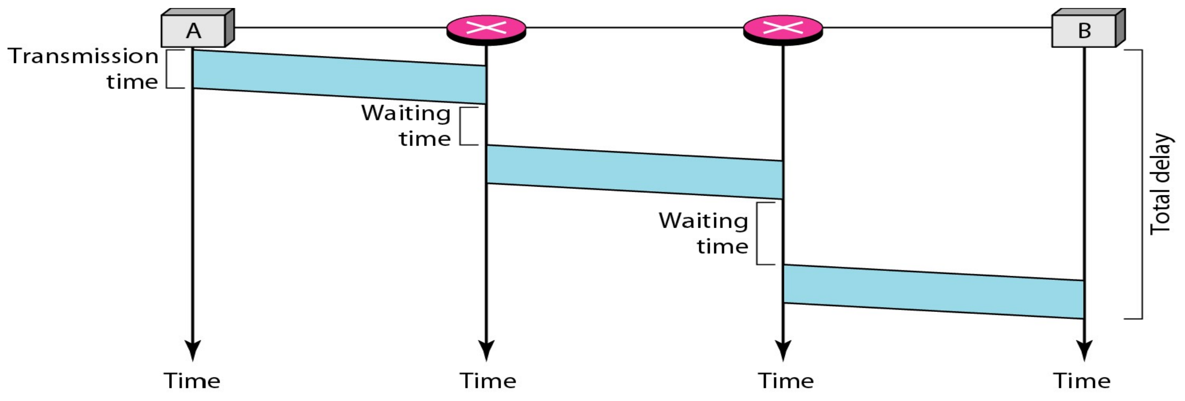
Efficiency:

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

Delay:

Total delay=3T+3T+W1+W2

Delay in a datagram network: GREATER

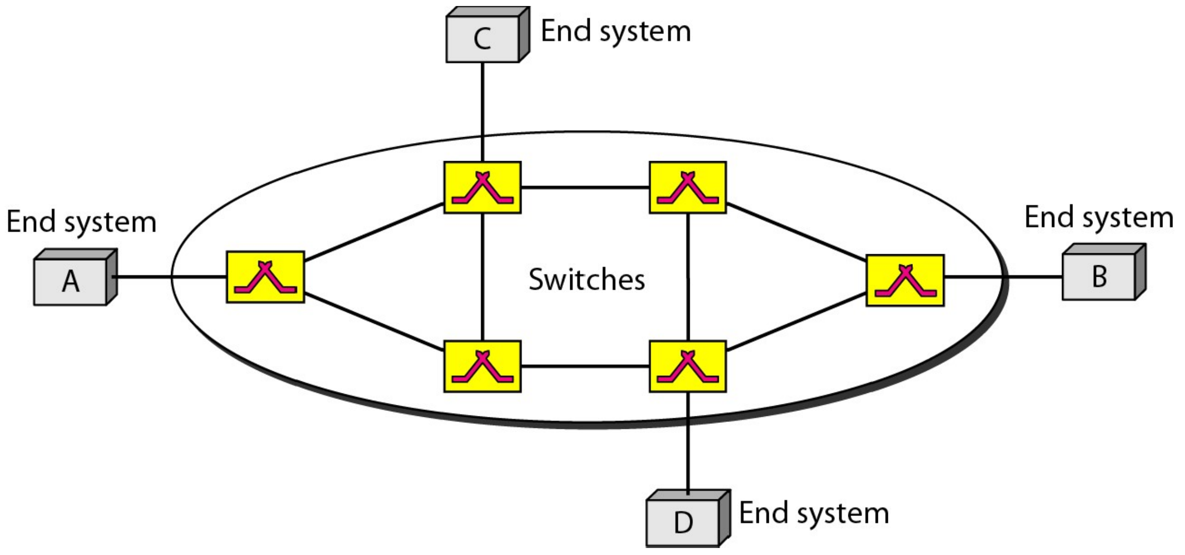


Switching in the Internet is done by using the datagram approach to packet switching at the network layer.

VIRTUAL-CIRCUIT NETWORKS:

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

- 3 phases same as circuit- switched networks.
- Resources can be allocated during setup phase as in circuit-switched n/w, or on demand, as in a datagram n/w.
- Same as datagram network, data are packetized. Addresses (localized) need for intermediate switch to route the packet.
- All packets follow the same path (connection-oriented) same as circuit-switched network.
- Virtual circuit network implemented in the data link layer.



Addressing:

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

Global Addressing:

A source or a destination needs to have a global address: an address that can be unique in the scope of the network or internationally if the network is part of an international network.

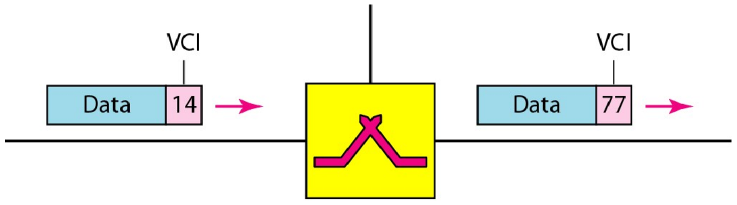
- A global address in virtual-circuit networks is used only to create a virtual-circuit identifier.

Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the virtual-circuit identifier (Vel).

- A vel , is a small number that has only switch scope; it is used by a frame between two switches.

- When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCI.



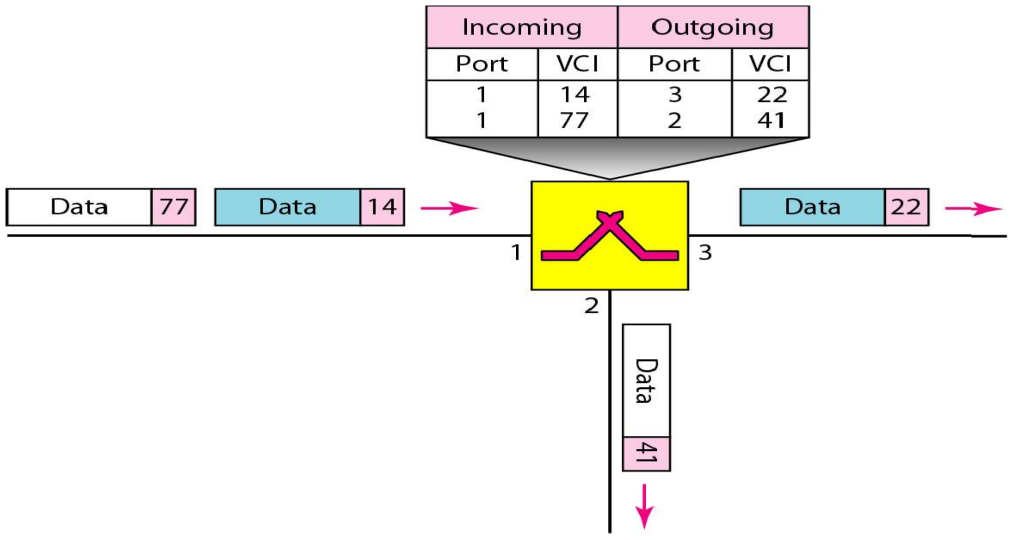
Three Phases

A source and destination need to go through three phases in a virtual-circuit network:

1. Setup
2. data transfer
3. Teardown.

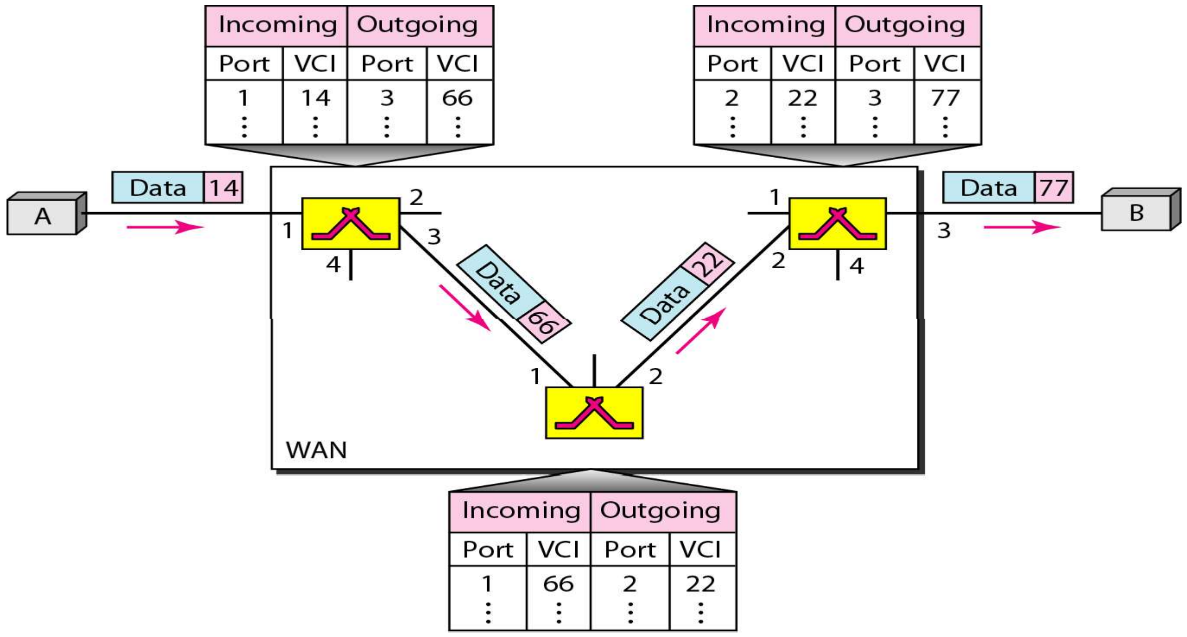
Data Transfer Phase:

- To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit.
- The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up.
- The data transfer phase is active until the source sends all its frames to the destination.
- The procedure at the switch is the same for each frame of a message. The process creates a virtual circuit, not a real circuit, between the source and destination.
- **Figure shows such a switch and its corresponding table.**



Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
1	77	2	41

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



Source-to-destination data transfer in a virtual-circuit network

Setup Phase: In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection.

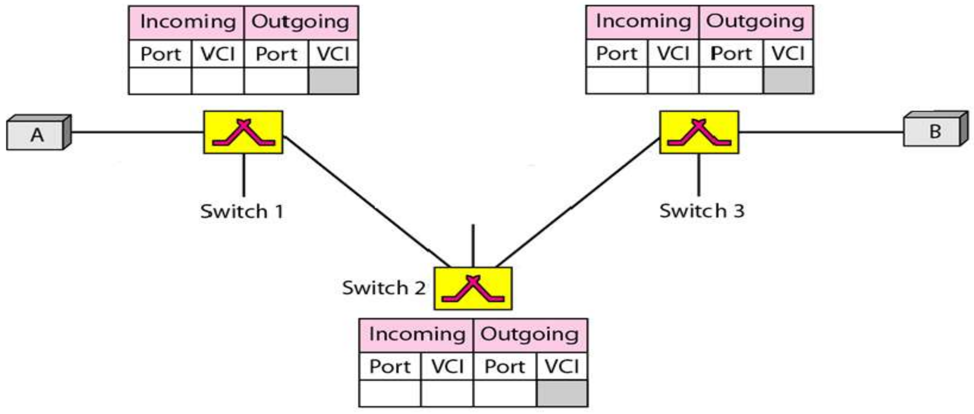
In the setup phase, a switch creates an entry for a virtual circuit. For ex, suppose source a needs to create a virtual circuit to B. Two steps are required: the setup request and the acknowledgment.

Setup Request

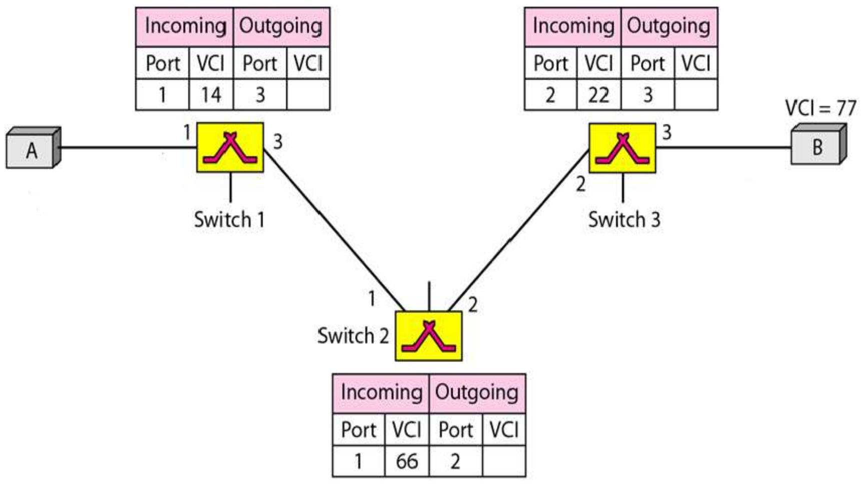
A setup request frame is sent from the source to the destination

Figure shows the process.

Setup request in a virtual-circuit network:



Setup acknowledgment in a virtual-circuit network:



Acknowledgment:

A special frame, called the acknowledgment frame, completes the entries in the switching tables.

Figure shows the process.

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

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

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

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

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

Teardown Phase:

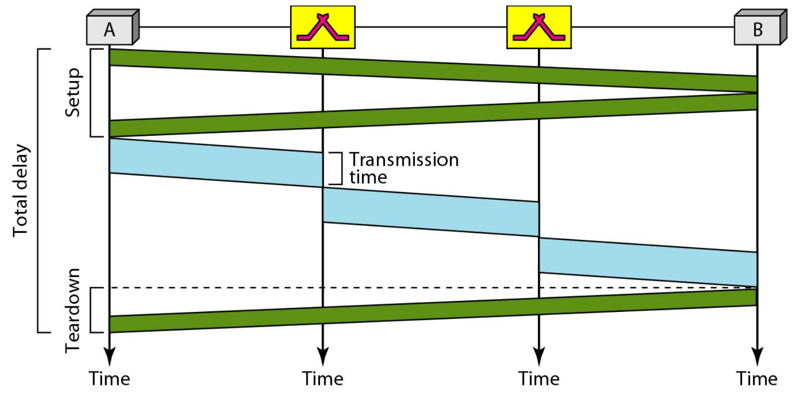
In this phase, source A, after sending all frames to B, sends a special frame called a teardown *request.* Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

Efficiency:

As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different de ays.

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

Delay:



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

10. Explain about DSL?

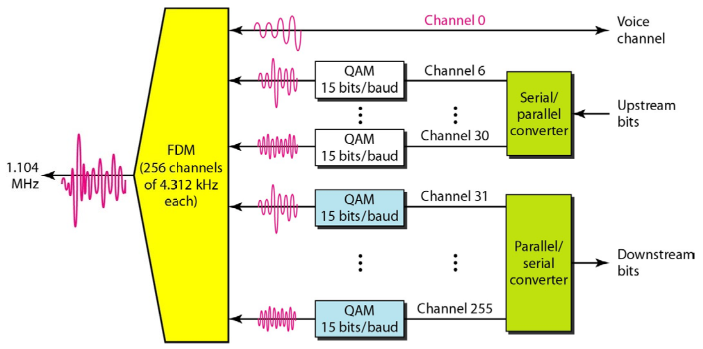
Digital subscriber line (DSL)

- DSL technology is one of the most promising for supporting high-speed digital communication over the existing local loops.
- After traditional modems reached their peak data rate, telephone companies developed another technology, DSL, to provide higher-speed access to the Internet.

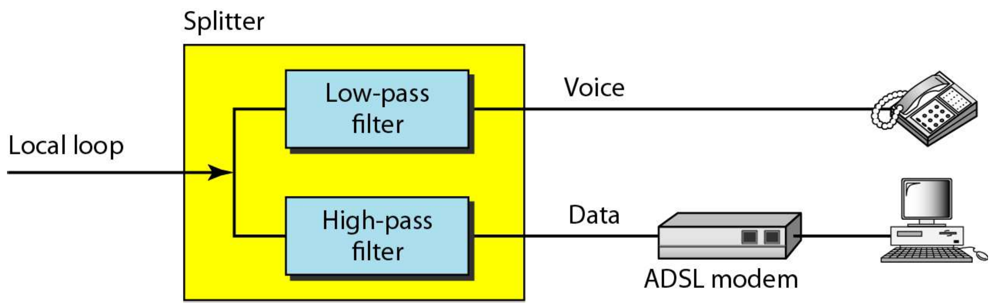
ADSL (Asymmetric DSL):

- ADSL is an asymmetric communication technology designed for residential users; it is not suitable for businesses.
- The existing local loops can handle bandwidths up to 1.1 MHz
- ADSL is an adaptive technology. The system uses a data rate based on the condition of the local loop line.

Discrete multitone technique:



ADSL modem:



Telephone Company Site: DSLAM

At the telephone company site, the situation is different. Instead of an ADSL modem, a device called a digital subscriber line access multiplexer (DSLAM) is installed that functions similarly.

ADSL Lite:

The installation of splitters at the border of the premises and the new wiring for the data line can be expensive and impractical enough to dissuade most subscribers. A new version of ADSL technology called ADSL Lite (or Universal ADSL or splitter less ADSL) is available for these subscribers. This technology allows an ASDL Lite modem to be plugged directly into a telephone jack and connected to the computer

- The splitting is done at the telephone company
- ADSL Lite uses 256 DMT carriers with 8-bit modulation
- SECTION 9.3 DIGITAL SUBSCRIBER LINE 255(instead of 15-bit)
- It can provide a maximum down- stream data rate of 1.5 Mbps and an upstream data rate of512 kbps.

HDSL (High-bit-rate DSL):

- The HDSL was designed as an alternative to the T-1line (1.544 Mbps). The T-1line uses alternate mark inversion (AMI) encoding, which is very susceptible to attenuation at high frequencies.
- This limits the length of a T-l line to 3200 ft (1 km). For longer distances, a repeater is necessary, which means increased costs. HDSL uses 2B1Q encoding (see Chapter 4), which is less A data susceptible to attenuation.
- Rate of 1.544 Mbps (sometimes up to 2 Mbps) can be achieved without repeaters up to a distance of 12,000 ft (3.86 km).
- HDSL uses two twisted pairs (one pair for each direction) to achieve full-duplex transmission.

SDSL (Symmetric DSL):

- The symmetric digital subscriber line (SDSL) is a one twisted-pair version of HDSL.
- It provides full-duplex symmetric communication supporting up to 768 kbps in each direction. SDSL, which provides symmetric communication, can be considered an alternative to ADSL.
- it is not suitable for businesses that send and receive data in large volumes in both directions.

VDSL (Very high-bit-rate DSL):

- The VDSL an alternative approach that is similar to ADSL, uses coaxial, fiber-optic, or twisted-pair cable for short distances.
- The modulating technique is DMT. It provides a range of bit rates (25 to 55 Mbps) for upstream communication at distances of3000 to 10,000 ft.
- The downstream rate is nor- mally 3.2 Mbps.

Summary of DSL technologies:

Technology	Downstream Rate	Upstream Rate	Distance (ft)	Twisted Pairs	Line Code
ADSL	1.5–6.1 Mbps	16–640 kbps	12,000	1	DMT
ADSL Lite	1.5 Mbps	500 kbps	18,000	1	DMT
HDSL	1.5–2.0 Mbps	1.5–2.0 Mbps	12,000	2	2B1Q
SDSL	768 kbps	768 kbps	12,000	1	2B1Q
VDSL	25–55 Mbps	3.2 Mbps	3000–10,000	1	DMT

11. Explain following Error control methods

d. **Hamming code**

e. **CRC**

f. **Checksum**

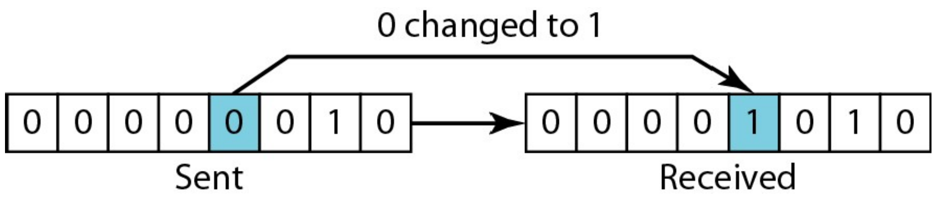
Ans: Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted.

- Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting errors.

Types of Errors:

Single-Bit Error:

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



(ASCII *STX)* ***(ASCII LF)***

Burst Error:

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

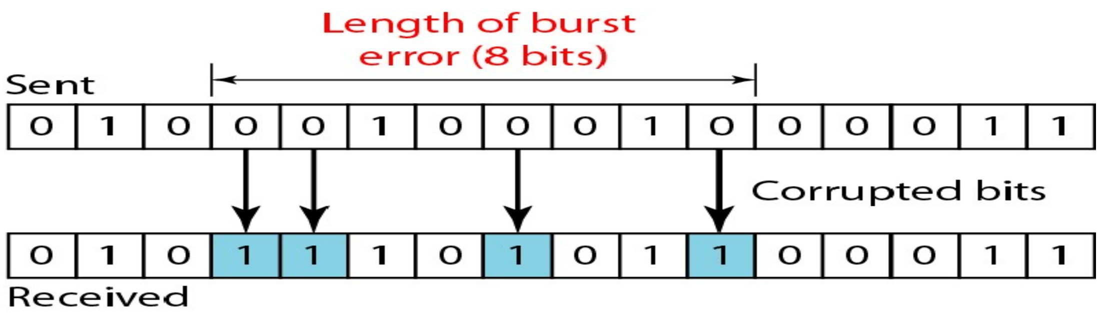
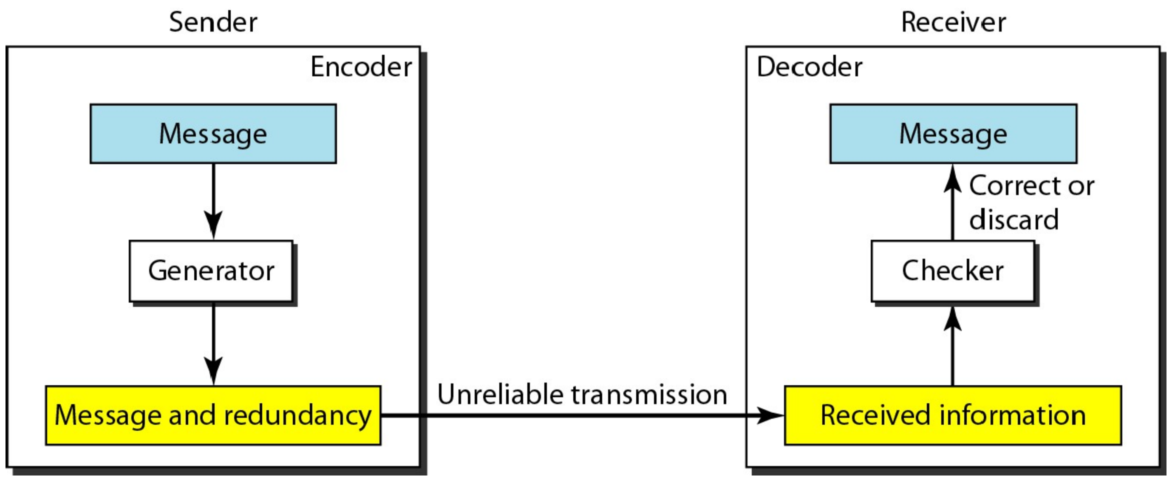


Fig: ***Burst error of length 8***

Redundancy:

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

The structure of encoder and decoder



Modular Arithmetic:

- In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus N. We then use only the integers 0 to N - I, inclusive.
- In modulo-N arithmetic, we use only the integers in the range 0 to N - 1, inclusive.
- Addition and subtraction in modulo arithmetic are simple.

Modulo-2 Arithmetic:

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

Adding: Subtracting:

0+0=0 0-0=0

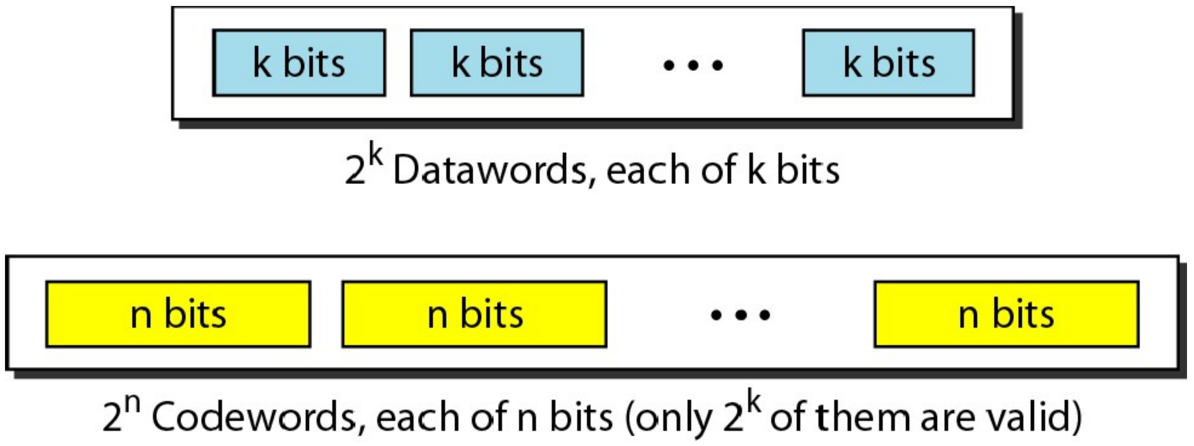
0+1=1 0-1=1

1+0=1 1-0=1

1+1=0 1-1=0

BLOCK CODING:

In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called code words.


k bits
k bits
• • •
k bits
2^k Datawords, each of k bits
n bits
n bits
• • •
n bits
2^n Codewords, each of n bits (only 2^k of them are valid)

Data words and code words in block coding

<u>Hamming code:</u>

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

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

1. The Hamming distance d (000, 011) is 2 because
$000 \oplus 011$ is 011 (two 1s)

2. The Hamming distance d (10101, 11110) is 3 because

$10101 \oplus 11110$ is 01011 (three 1s)

- **The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.**
- **To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.**
- **To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.**

<u>CRC:</u>

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a code word is cyclically shifted (rotated), the result is another code word.

For example,if 1011000 is a code word and we cyclically left-shift, then 0110001 is also a code word.

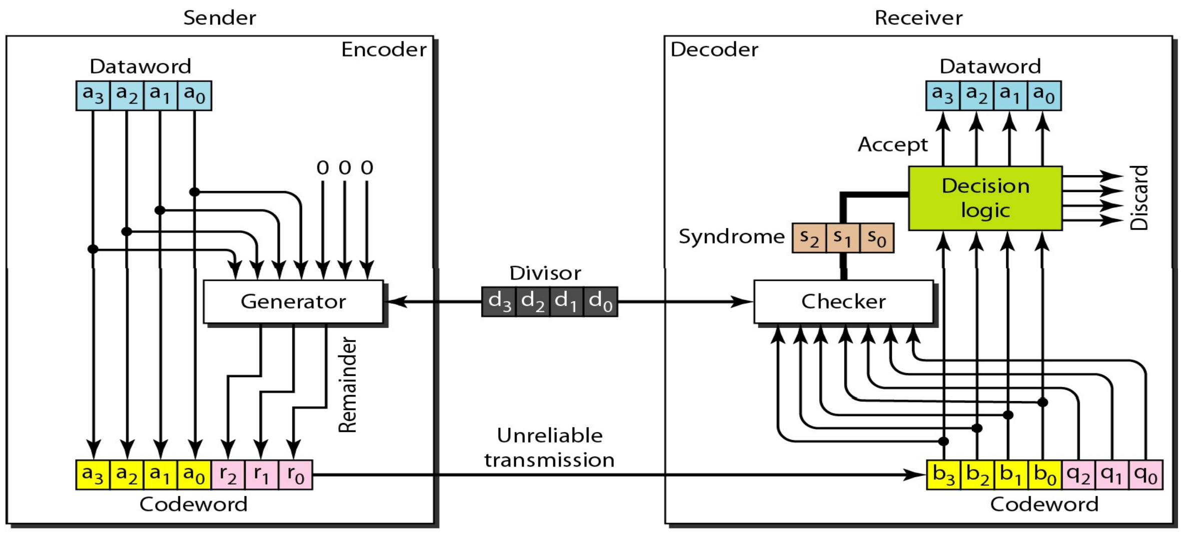
In this case, if we call the bits in the first word ao to a6' and the bits in the second word b0 to b6, we can shift the bits by using the following:

b1=a0 b2=a1 b3=a2 b4=a3 b5=a4 b6=a5 b0=a6

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.***A CRC code with C (7, 4)***

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

CRC encoder and decoder:



In the encoder, the data word has k bits (4 here); the code word has n bits (7 here).

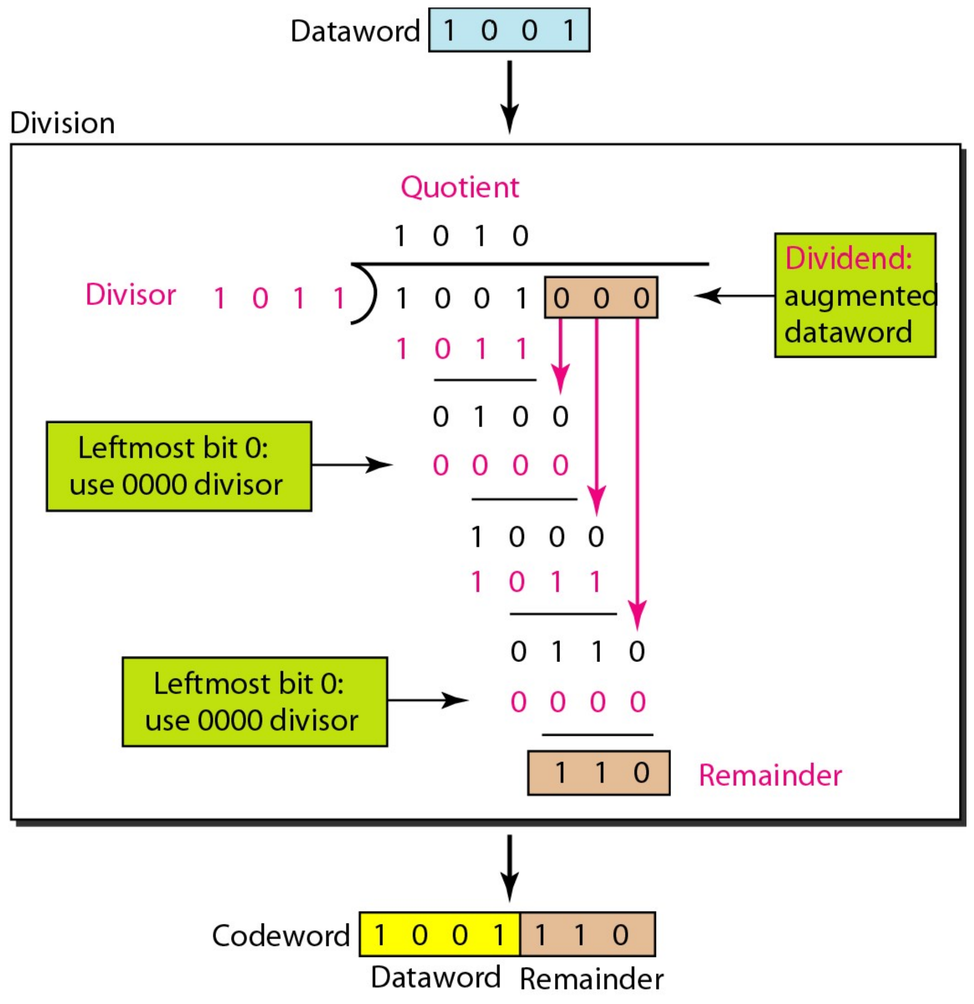
The size of the data word is augmented by adding n - k (3 here) 0s to the right-hand side of the word. The n- bit result is fed into the generator. The generator uses a divisor of size n - k + I (4 here), predefined and agreed upon. The generator divides the augmented data word by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r2rl ro) is appended to the data word to create the code word.

The decoder receives the possibly corrupted code word. A copy of all n bits is fed to the checker which is a replica of the generator. The remainder produced by the checker is a syndrome of n - k (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all as, the 4 leftmost bits of the code word are accepted as the data word (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

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

Division in CRC encoder:

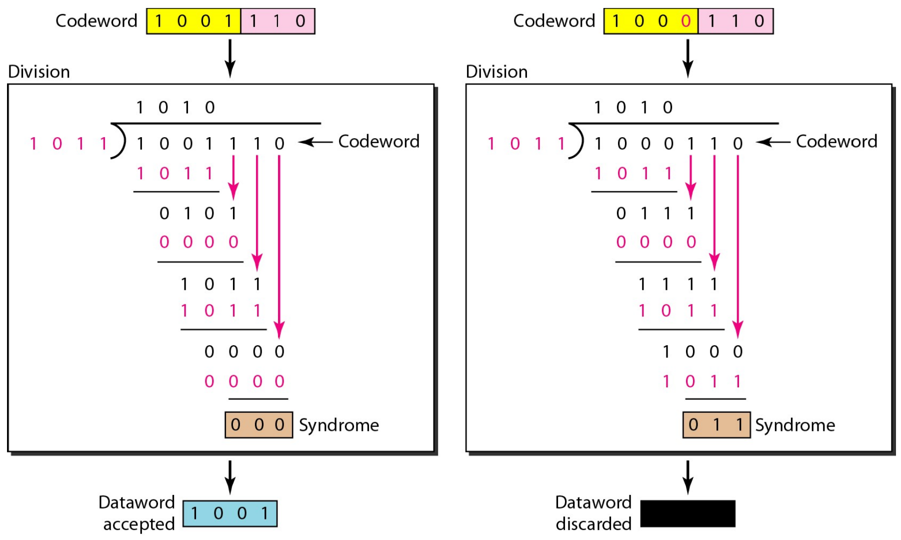


Decoder

The code word can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error; the data word is separated from the received code word and accepted. Otherwise, everything is discarded. Figure 10.16

shows two cases: The left hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all 0s (it is OIl).

Division in the CRC decoder for two cases:



Augmented Dataword

In our paper-and-pencil division process in Figure 10.15, we show the augmented data word as fixed in position with the divisor bits shifting to the right, 1 bit in each step.

The divisor bits are aligned with the appropriate part of the augmented data word. Now that our divisor is fixed, we need instead to shift the bits of the augmented data word to the left (opposite direction) to align the divisor bits with the appropriate part. There is no need to store the augmented data word bits.

Remainder

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

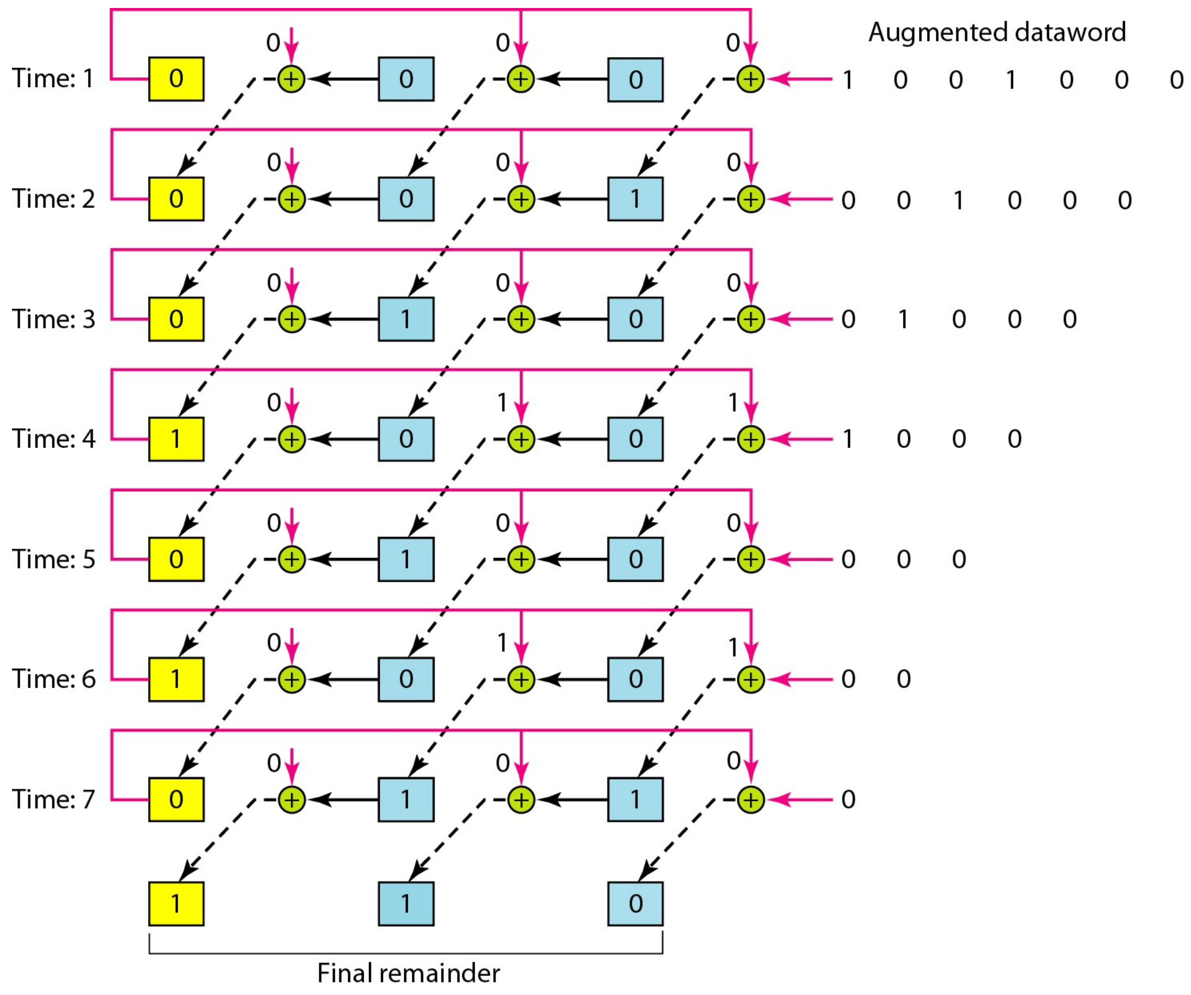
1. We assume that the remainder is originally all 0s (000 in our example).

2. At each time click (arrival of 1 bit from an augmented data word), we repeat the following two actions:

a. We use the leftmost bit to make a decision about the divisor (011 or 000).

b. The other 2 bits of the remainder and the next bit from the augmented data word (total of 3 bits) are XORed with the 3-bit divisor to create the next remainder.

Simulation of division in CRC encoder:



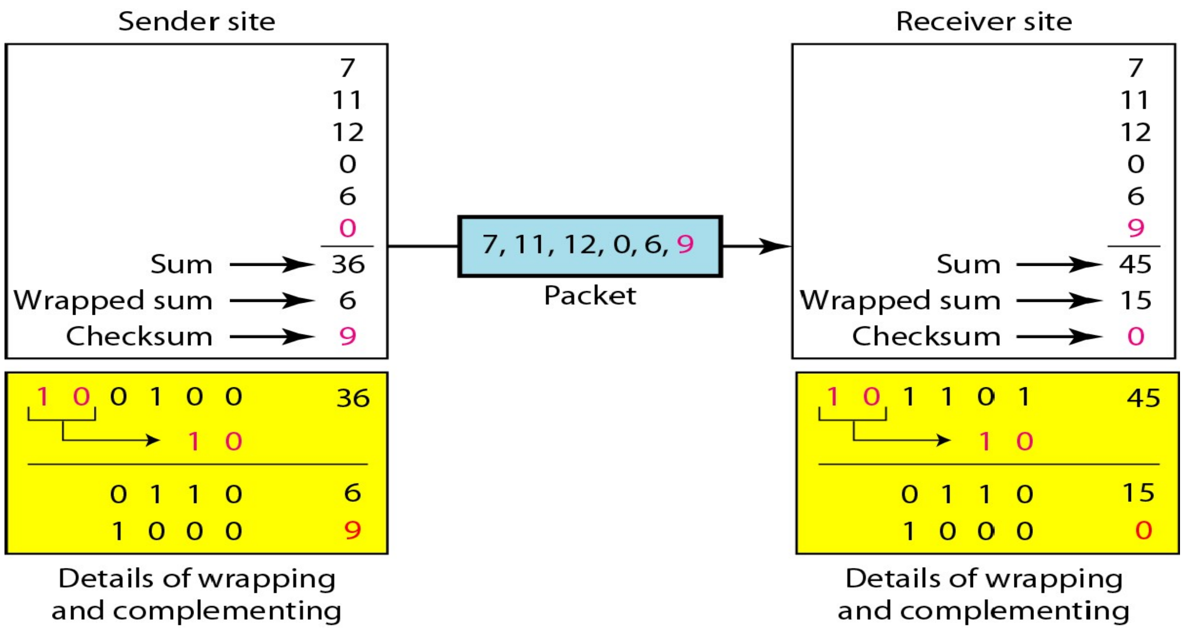
CHECKSUM:

The checksum is used in the Internet by several protocols although not at the data link layer.

Implementation of checksum:

The sender initializes the checksum to 0 and adds all data items and the checksum. 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6.

The sum is then complemented, resulting in the checksum value 9 (15 − 6 = 9).



Example:

Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using one's complement

```
10101001
                 00111001
                 ------------
Sum              11100010

Checksum          00011101
```

The pattern sent is 10101001 00111001 00011101

Now suppose the receiver receives the pattern sent in Example 7 and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

```
                      10101001

                      00111001

                     00011101

                      ------------

Sum                  11111111
```

Complement 00000000 means that the pattern is OK.

Now suppose there is a burst error of length 5 that affects 4 bits.

10101111 11111001 00011101

When the receiver adds the three sections, it gets

```
                 10101111
                 11111001
                 00011101
                 ------------
Partial Sum     111000101
Carry                   1
Sum              11000110
Complement       00111001    the pattern is corrupted.
```

Internet Checksum:

Sender site:

1. The message is divided into 16-bit words.

2. The value of the checksum word is set to 0.

3. All words including the checksum are added using one's complement addition.

4. The sum is complemented and becomes the checksum.

5. The checksum is sent with the data.

Receiver site:

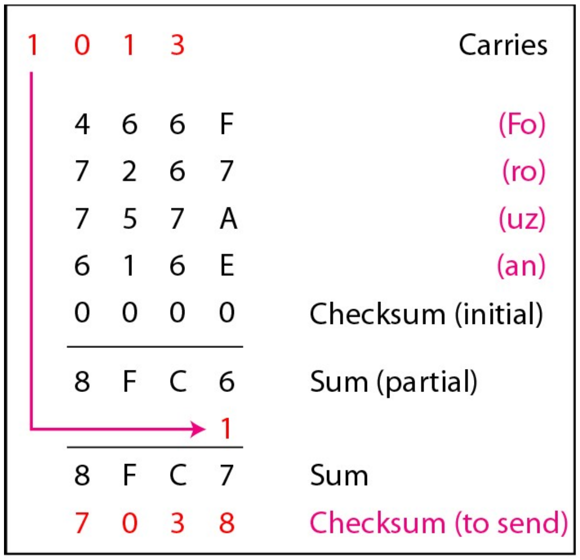
1. The message (including checksum) is divided into 16-bit words.

2. All words are added using one's complement addition.

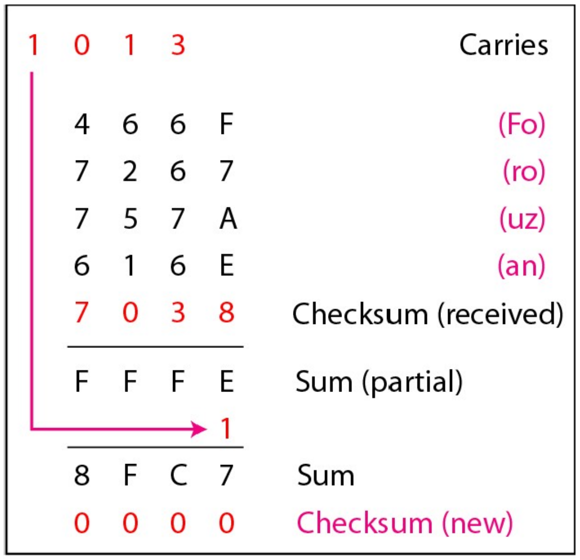
3. The sum is complemented and becomes the new checksum.

4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Example:

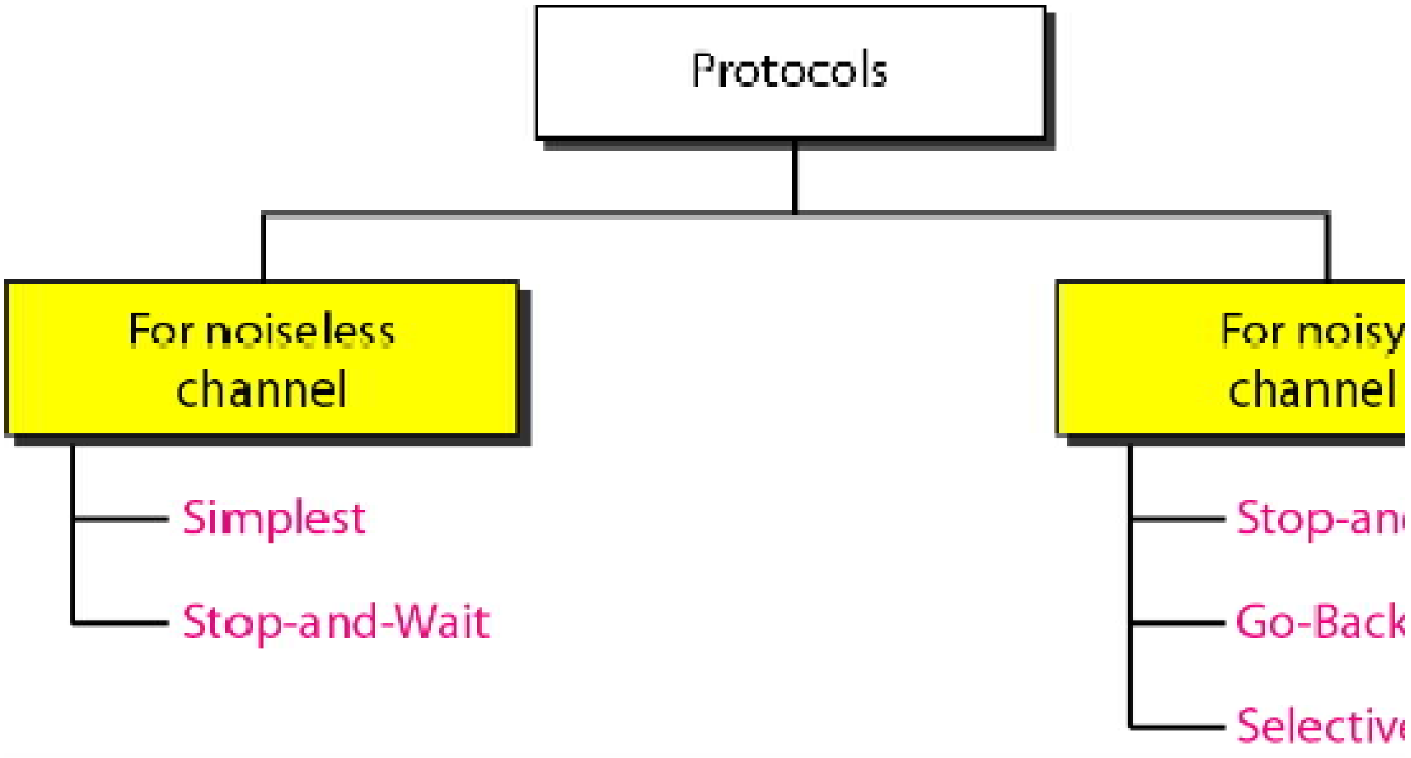


a. Checksum at the sender site



a. Checksum at the receiver site

12)Explain flow and error control protocols of noisy channels?



Stop-and-Wait Automatic Repeat Request:

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-andWait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol.
Let us see how this protocol detects and corrects errors.
To detect and correct corrupted frames, we need to add redundancy bits to our data frame .When the frame arrives at the receiver site, it is checked and if
it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.
Lost frames are more difficult to handle than corrupted ones. In other protocols there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.
The comlpted and lost frames need to be resent in this protocol error, At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of
the same frame can be in the network.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.
Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. The ACK frame for this protocol has a sequence number field.
In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

Sequence Numbers:
The protocol specifies that frames need to be numbered. This is done
by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.
One important consideration is the range of the sequence numbers. Since we want to minimize the frame size, we look for the smallest range that provides unambiguous communication. The sequence numbers of course can wrap around.
assume that the sender has sent the frame numbered *x*. Three things can happen.
1. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment.
The acknowledgment arrives at the sender site, causing the sender to send
the next frame numbered $x + 1$.
2. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost. The sender resends the frame (numbered *x)* after the time-out. Note that the frame here is a duplicate. The receiver
can recognize this fact because it expects frame x + I but frame x was received.
3. The frame is corrupted or never arrives at the receiver site; the sender resends the frame (numbered *x)* after the time-out.

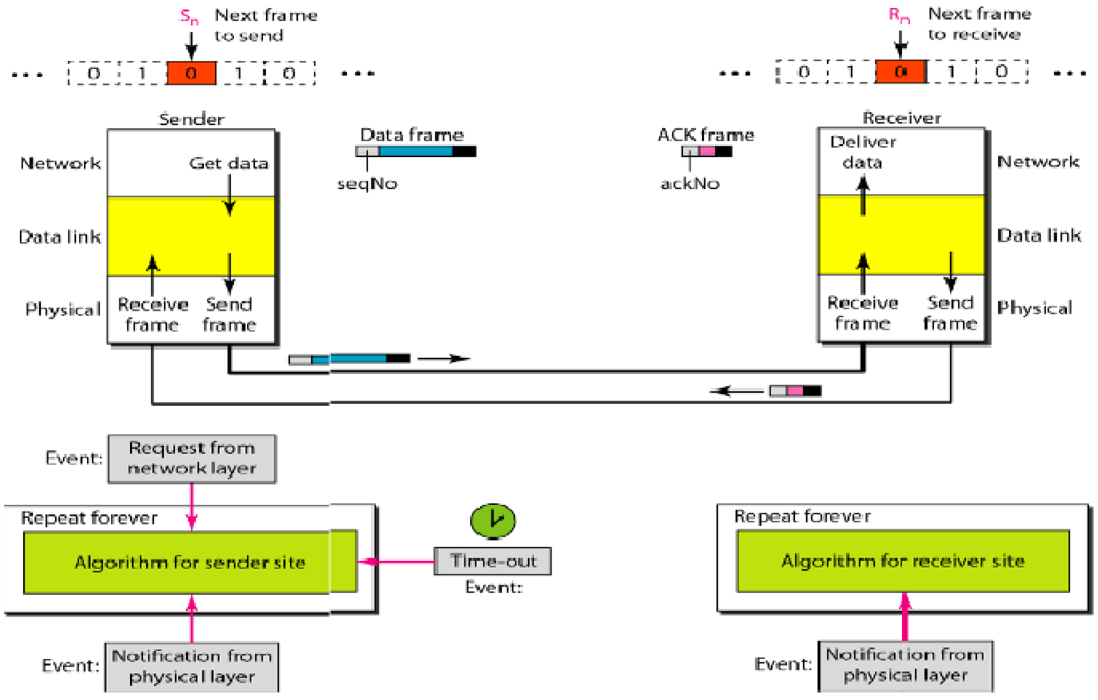
Acknowledgment Numbers

The acknowledgment numbers always announce the sequence

number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an
ACK frame with acknowledgment 0 (meaning frame 0 is expected).

Design

The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number). The sender has a control variable, which we call *Sn* (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).



The receiver has a control variable, which we call *Rn* (receiver, next frame expected),

that holds the number of the next frame expected. When a frame is sent, the value of *Sn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of *Rn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable *Sn* points to the slot that matches the sequence

number of the frame that has been sent, but not acknowledged; *Rn* points to the slot that matches the sequence number of the expected frame.

Pipelining

In networking and in other areas, a task is often begun before the previous task has ended.

This is known as pipelining. There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next

frame can be sent. However, pipelining does apply to our next two protocols because several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment.

The first is called Go-Back-N Automatic Repeat Request (the rationale for the name will become clear later). In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sliding Window:

In this protocol ,the sliding window defines the range of sequence numbers that is the concern of the sender and receiver

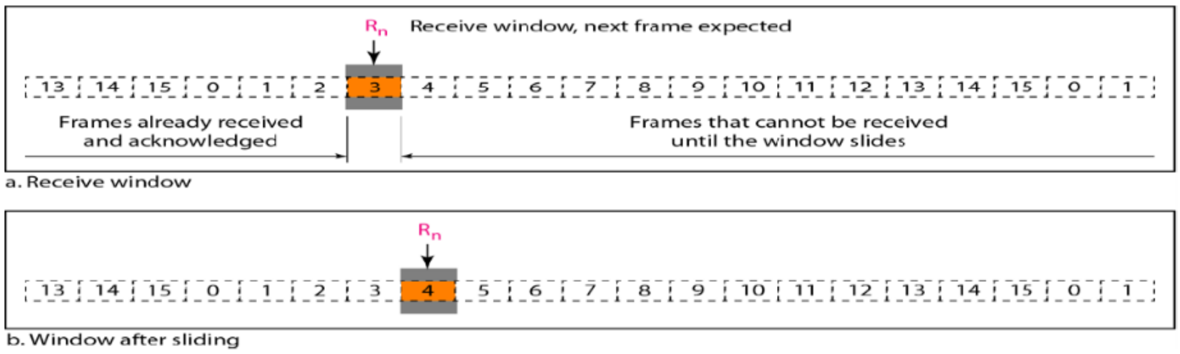
The sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window.

The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is $2m – 1$.

Figure shows a sliding window of size 15 *(m* =4).

The window at any time divides the possible sequence numbers into four regions.

The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender n eds to wait to find out if these frames have been received or were lost. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network



The receive window is an abstract concept defining an imaginary box of size 1 with one single variable *Rn*• The window slides when a correct frame has arrived; sliding occurs one slot at a time. we need only one variable *Rn* (receive window, next frame expected) to define this abstraction. The sequence numbers to the

left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this
window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of*Rn* is accepted and acknowledged.
The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.

Timers

Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

Acknowledgment
The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides

Selective Repeat Automatic Repeat Request:

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames.
This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective RepeatARQ.

Windows :

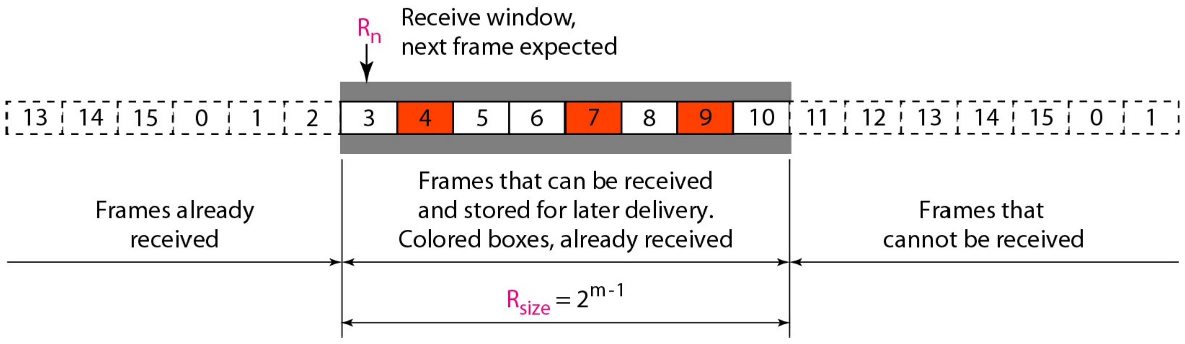
The Selective Repeat Protocol also uses two windows: a send window and a receive window.
However, there are differences between the windows in this protocol and the ones in Go-Back-N. First, the size of the send window is much smaller; it is *2m- I* . The reason for this will be discussed later. Second, the receive window is the same size as the send window. The send window maximum size can be *2m- I* . For example, if *m* = 4, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the *Go-Back-N* Protocol). The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this.
The protocol uses the same variables as we discussed for Go-Back-N.

Selective Repeat Automatic Repeat Request:-

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames.

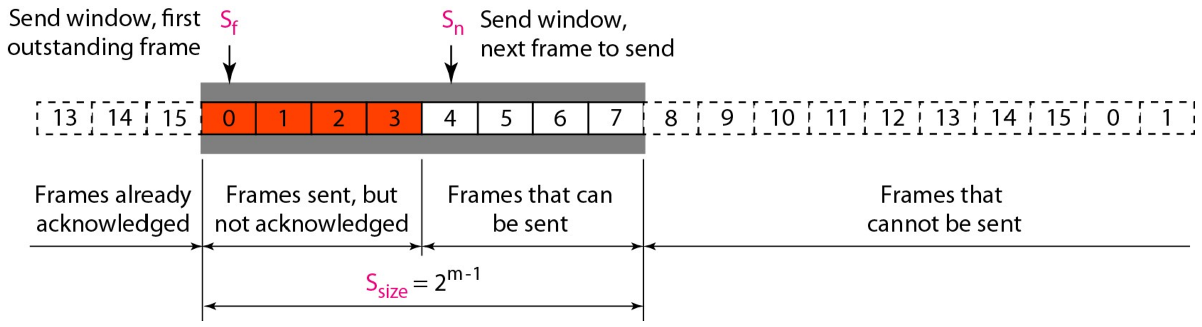
This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective RepeatARQ.
It is more efficient for noisy links, but the processing at the receiver is more complex.

Windows
The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are differences between the windows in this protocol and the ones in Go-Back-N. First, the size of the send window is much smaller; it is *2m- I* . The reason for this will be discussed later. Second, the receive window is the same size as the send window.
The send window maximum size can be *2m- I* . For example, if *m* = 4, the
sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the *Go-Back-N* Protocol). The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this.
The protocol uses the same variables as we discussed for Go-Back-N. We show the Selective Repeat send window in Figure to emphasize the size.



R_n
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

Sending:
Receiving:



Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

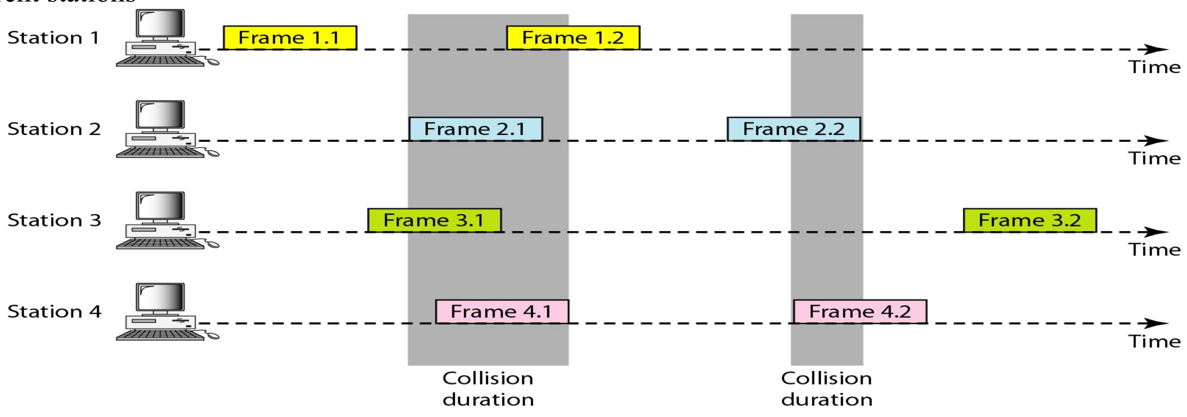
13. What are various random access protocols ?

In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is The random access methods we study in this chapter have evolved from a very interesting protocol known as ALOHA
ALOHA :

sends data, another station may attempt to do so at the same time. The data from the two stations collide and become The earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station garbled .

Pure ALOHA :

The original ALOHA protocol is called pure ALOHA. This is a simple, but elegant
protocol. The idea is that each station sends a frame whenever it has a frame to send.
However, since there is only one channel to share, there is the possibility of collision between frames from different stations



There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.3 shows that only two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver.
When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time *TB.*

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts Kmax' a station must give up and try later. Figure shows the procedure for pure ALOHA based on the above strategy.

There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.3 shows that only two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out

period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

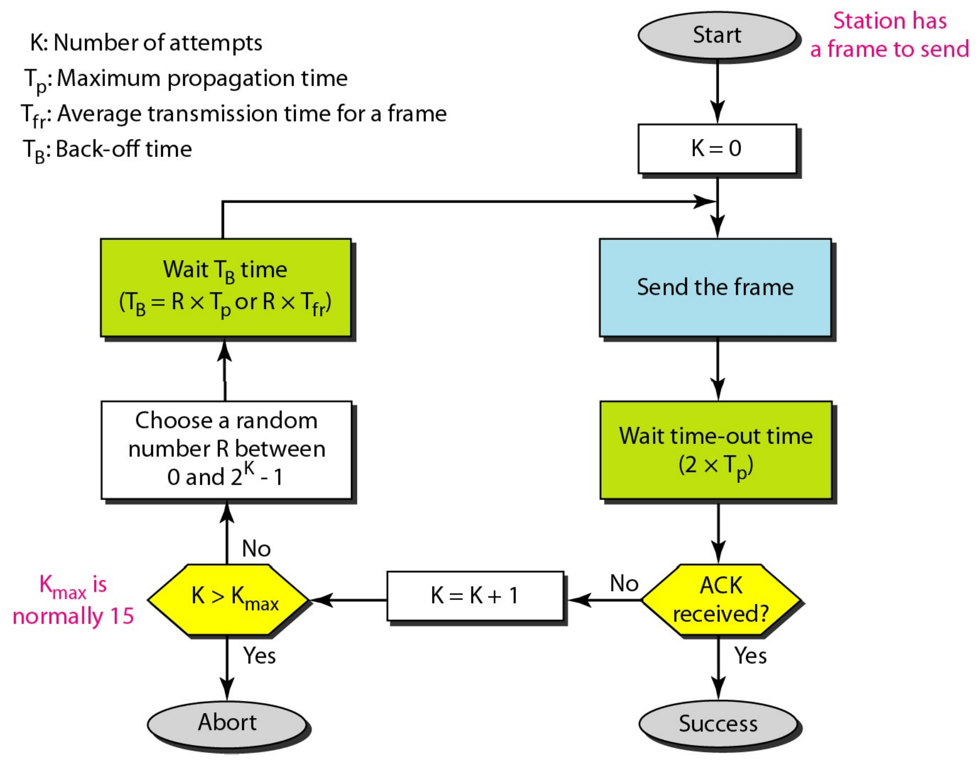
A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time *TB.*

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts Kmax' a station must give up and try later. Figure 12.4 shows the procedure for pure ALOHA based on the above strategy.

There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.3 shows that only two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame. A collision involves two or more stations. If all these stations try to resend their frames after the time- out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time *TB.*

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts Kmax' a station must give up and try later. Figure 12.4 shows the procedure for pure ALOHA based on the above strategy.

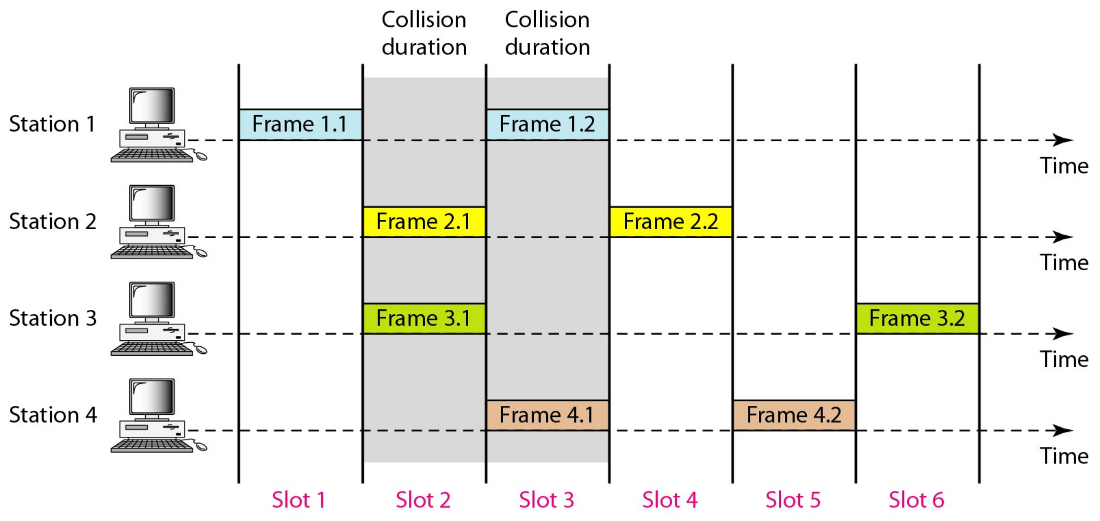

K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Start
Station has a frame to send
K = 0
Send the frame
Wait time-out time ($2 \times T_p$)
ACK received?
Yes
Success
No
K = K + 1
$K > K_{max}$
K_{max} is normally 15
Yes
Abort
No
Choose a random number R between 0 and $2^K - 1$
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)

The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations (2 x *Tp)'* The back-off time *TB* is a random value that normally depends on *K* (the number of attempted unsuccessful transmissions). The formula for *TB* depends on the implementation. One common formula is the **binary exponential back-off.** In this method, for each retransmission, a multiplier in the range 0 to *2K* - 1 is randomly chosenand multiplied by *Tp* (maximum propagation time) or *Trr* (the average time required to send out a frame) to find *TB'* Note that in this procedure, the range of the random numbers increases after each collision. The value of *Kmax* is usually chosen as 15.

Slotted ALOHA :

Pure ALOHA has a vulnerable time of 2 x *Tfr* . This is so because there is no rule thatdefines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.

In slotted ALOHA we divide the time into slots of Tfr s and force the station to send only at the beginning of the time slot



Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to *Tfr* Figure 12.7 shows the situation.

Figure 12.7 shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.
SlottedALOHA vulnerable time = Tfr

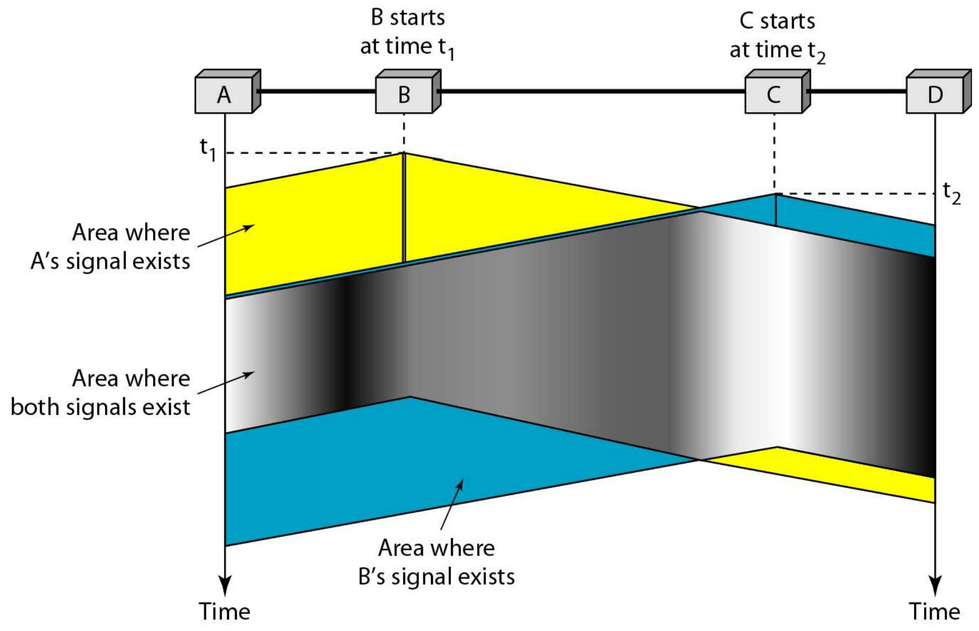
Throughput It can be proved that the average number of successful transmissions for slotted ALOHA is S = G x *e-G*. The maximum throughput Smax is 0.368, when G = 1. In other words, if a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. This result can be expected because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

Carrier Sense Multiple Access (CSMA) :

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk." CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 12.8, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station.

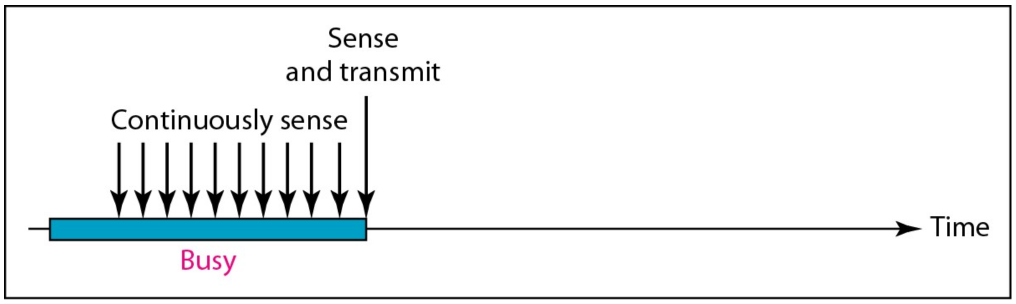
Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to *Tfr* Figure 12.7 shows the situation.


B starts at time t_1
C starts at time t_2
A
B
C
D
t_1
t_2
Area where A's signal exists
Area where both signals exist
Area where B's signal exists
Time
Time

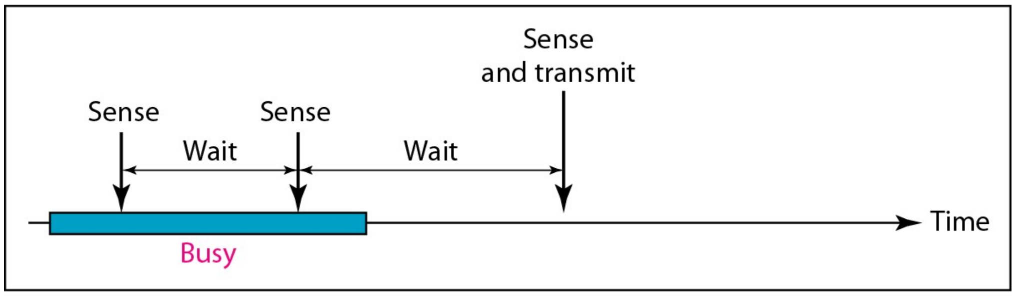
and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.
At time *tI'* station B senses the medium and finds it idle, so it sends a frame. At time *t2 (t2> tI)'* station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time
The vulnerable time for CSMA is the propagation time *Tp* . This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. case. The leftmost station A sends a frame at time *tl'* which reaches the rightmost station D at time *tl* + *Tp* . The gray area shows the vulnerable area in time and space.

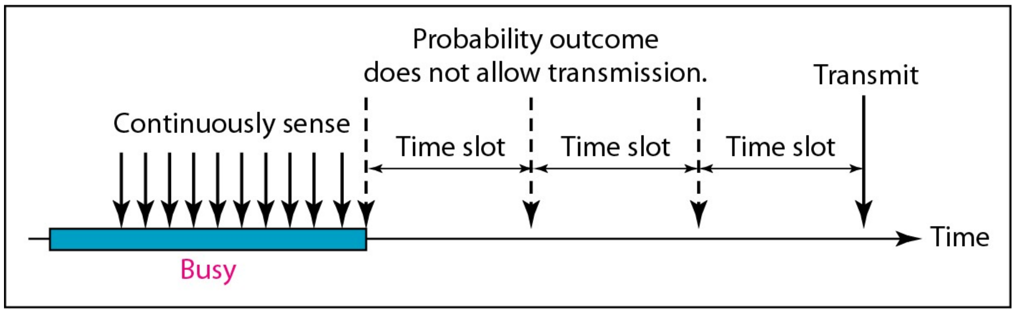
Persistence Methods
What should a station do if the channel is busy? What should a station do if the channel
is idle? Three methods have been devised to answer these questions: the I-persistent method, the nonpersistent method, and the p-persistent method.



a. 1-persistent



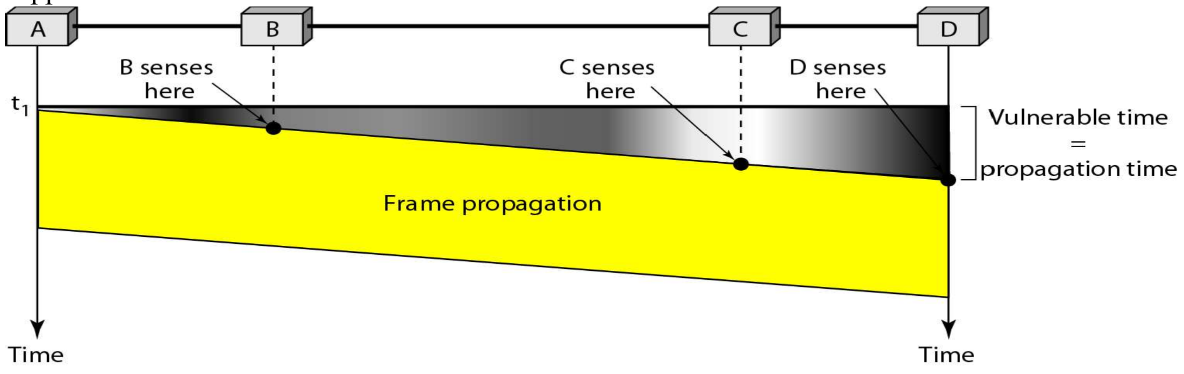
b. Nonpersistent



c. p-persistent

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) :

The CSMA method does not specify the procedure following a collision. Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.
In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.
To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In FIGURE stations A and C are involved in the collision.



Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) :

The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles.

However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection. We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance *(CSMAlCA)* was invented for this network. Collisions are avoided through the use of CSMAICA's three strategies: the interframe space, the contention window, and acknowledgments,

14. Explain about Ethernet MAC sub layer?

MAC Sublayer
In Standard Ethernet, the MAC sublayer governs the operation of the access method.

It also frames data received from the upper layer and passes them to the physical layer.

FRAME FORMAT

- *Preamble. The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The preamble is actually added at the physical layer and is not (formally) part of the frame.*

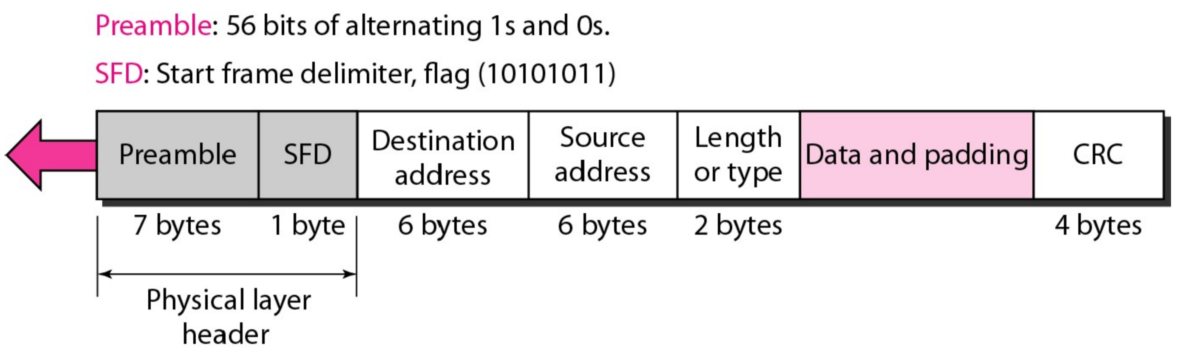
- *Start frame delimiter (SFD). The second field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.*

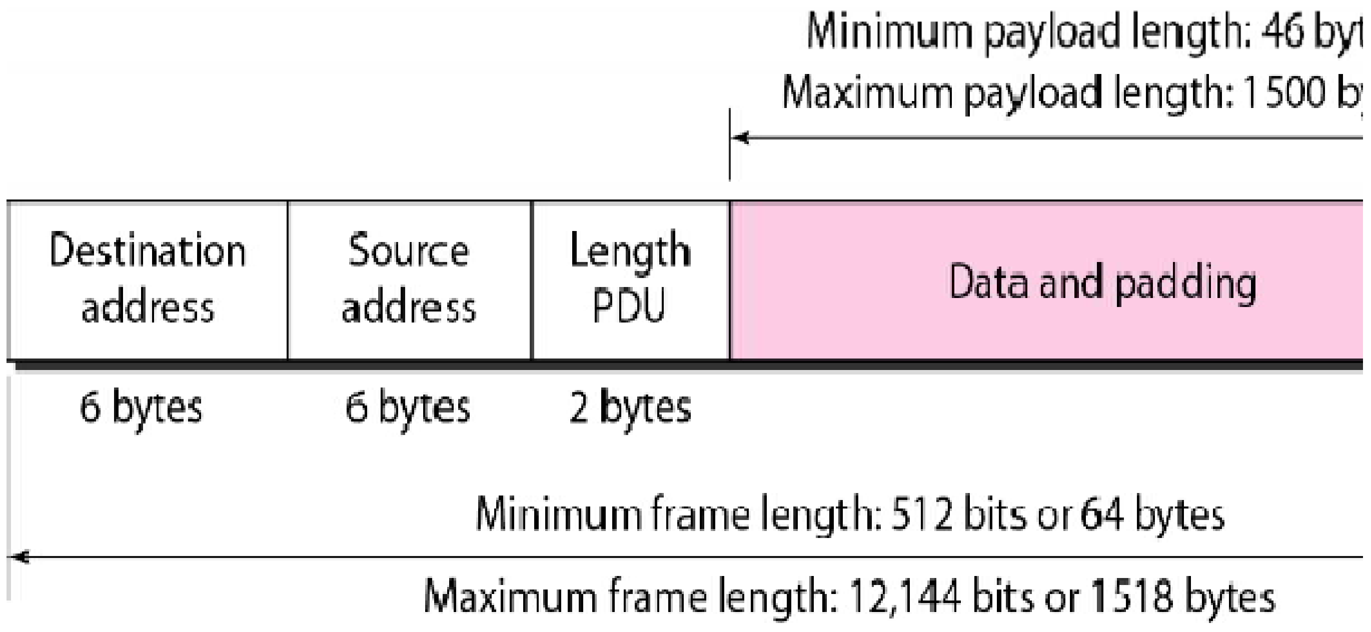
- *Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.*

- *Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet.*

- *Length or type. This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.*

- *Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.*
- *CRC. The last field contains error detection information, in this case a CRC-32.*




Minimum payload length: 46 byt
Maximum payload length: 1500 b
Destination address
Source address
Length PDU
Data and padding
6 bytes
6 bytes
2 bytes
Minimum frame length: 512 bits or 64 bytes
Maximum frame length: 12,144 bits or 1518 bytes

15. FRAME RELAY

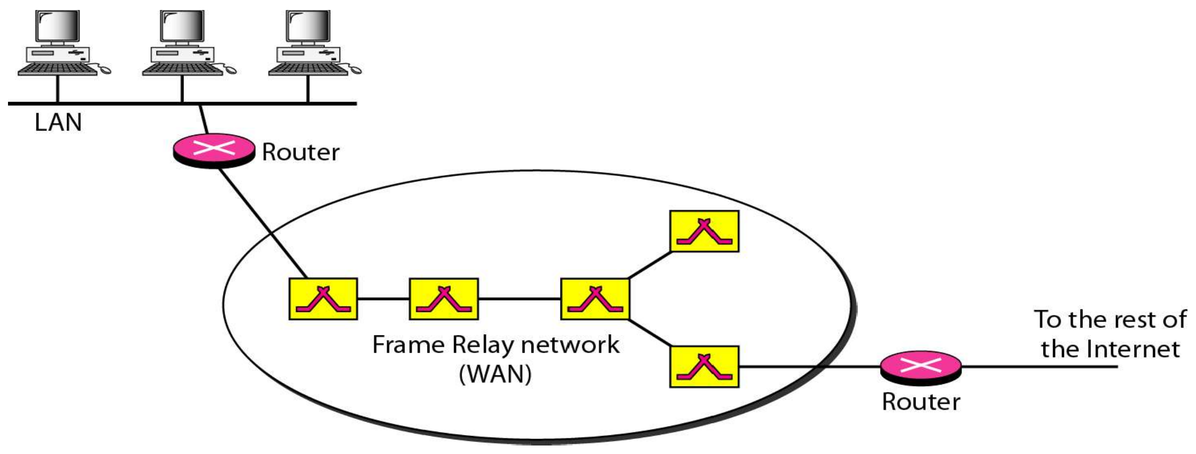
Frame Relay is a virtual-circuit wide-area network that was designed in response to demands for a new type of WAN in the late 1980s and early 1990s.

Originally designed for transport across Integrated Services Digital Network (ISDN) infrastructure, it may be used today in the context of many other network interfaces. Network providers commonly implement frame relay for voice (VoFR) and data as an encapsulation technique, used between local area networks (LANs) over a wide area network (WAN). Frame Relay is a wide area network with the following features:

- Frame Relay operates at a higher speed (1.544 Mbps and recently 44.376 Mbps). This means that it can easily be used instead of a mesh of T-I or T-3 lines.
- Frame Relay operates in just the physical and data link layers. This means it can easily be used as a backbone network to provide services to protocols that already have a network layer protocol, such as the Internet.
- Frame Relay allows bursty data.
- Frame Relay allows a frame size of 9000 bytes, which can accommodate all local area network frame sizes.
- Frame Relay is less expensive than other traditional WANs.
- Frame Relay has error detection at the data link layer only. There is no flow control or error control. There is not even a retransmission policy if a frame is damaged.
- Frame Relay was designed in this way to provide fast transmission capability for more reliable media and for those protocols that have flow and error control at the higher layers.

Architecture

Frame Relay provides permanent virtual circuits and switched virtual circuits.



Virtual Circuits

Frame Relay is a virtual circuit network. A virtual circuit in Frame Relay is identified by a number called a data link connection identifier (DLCI).

Permanent versus Switched Virtual Circuits

A source and a destination may choose to have a permanent virtual circuit (PVC). In this case, the connection setup is simple. The corresponding table entry is recorded for all switches by the administrator (remotely and electronically, of course). An outgoing DLCI is given to the source, and an incoming DLCI is given to the destination. PVC connections have two drawbacks. First, they are costly because two parties pay for the connection all the time even when it is not in use. Second, a connection is created from one source to one single destination. If a source needs connections with several destinations, it needs a PVC for each connection.

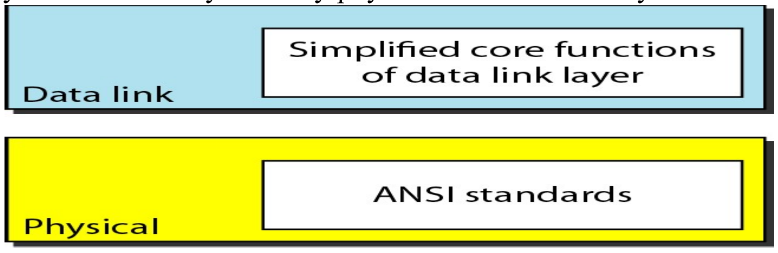
Switched virtual circuit (SVC). The SVC creates a temporary, short connection that exists only when data are being transferred between source and destination. An SVC requires establishing and terminating phases.

Switches

Each switch in a Frame Relay network has a table to route frames. The table matches an incoming port-DLCI combination with an outgoing port-DLCI combination as we described for general virtual-circuit networks in Chapter 8. The only difference is that VCIs are replaced by DLCIs.

Frame Relay Layers

Frame Relay layers. Frame Relay has only physical and data link Layers.



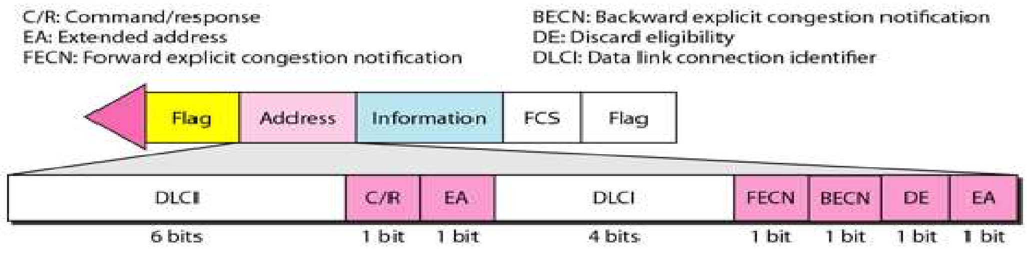
Physical Layer

No specific protocol is defined for the physical layer in Frame Relay. Instead, it is left to the implementer to use whatever is available. Frame Relay supports any of the protocols recognized by ANSI.

Data Link Layer

At the data link layer, Frame Relay uses a simple protocol that does not support flow or error control. It only has an error detection mechanism. The address field defines the DLCI as well as some bits used to control congestion.

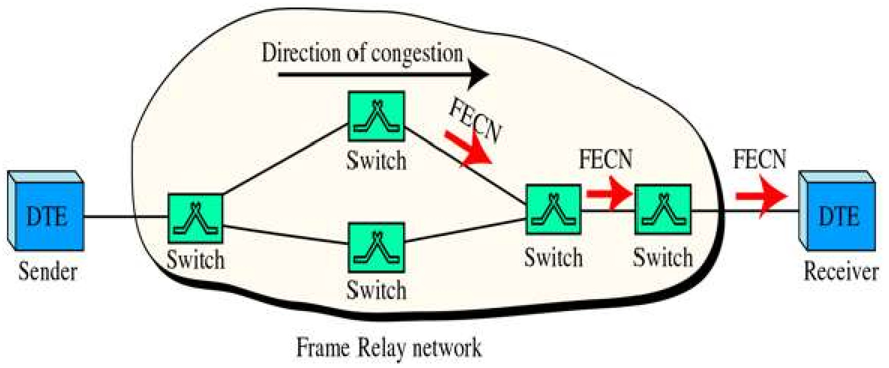
Frame Relay frame



The descriptions of the fields are as follows:

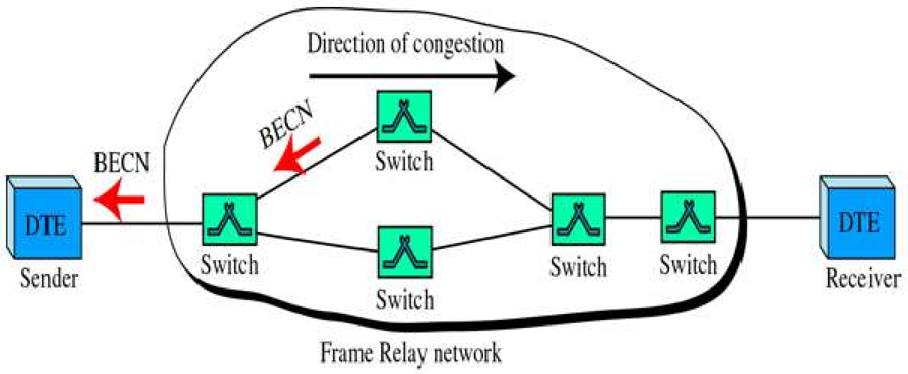
- **Address (DLCI) field**. The first 6 bits of the first byte makes up the first part of the DLCI. The second part of the DLCI uses the first 4 bits of the second byte. These bits are part of the lO-bit data link connection identifier defined by the standard.
- **Command/response (CIR)**. The command/response (C/R) bit is provided to allow upper layers to identify a frame as either a command or a response. It is not used by the Frame Relay protocol.
- **Extended address (EA)**. The extended address (EA) bit indicates whether the current byte is the final byte of the address. An EA of 0 means that another address byte is to follow.
- **Forward explicit congestion notification (FECN).** The forward explicit congestion notification (FECN) bit can be set by any switch to indicate that traffic is congested. This bit informs the destination that congestion has occurred. In this way, the destination knows that it should expect delay or a loss of packets.

FECN



- Backward explicit congestion notification (BECN). The backward explicit congestion notification (BECN) bit is set (in frames that travel in the other direction) to indicate a congestion problem in the network. This bit informs the sender that congestion has occurred.

BECN

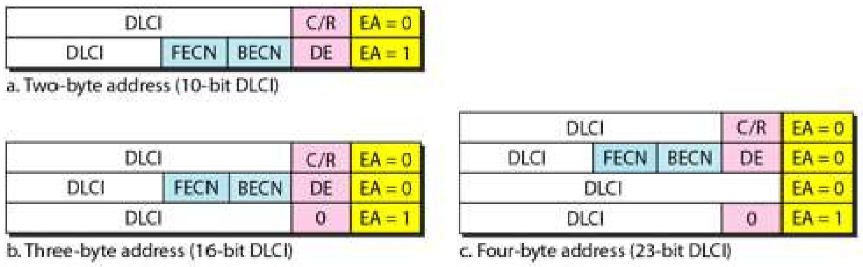


- Discard eligibility (DE). The discard eligibility (DE) bit indicates the priority level of the frame. When set (DE 1), this bit tells the network to discard this frame if there is congestion. This bit can be set either by the sender of the frames (user) or by any switch in the network.
- Frame Relay does not provide flow or error control. They must be provided by the upper-layer protocols.

Extended Address

To increase the range of DLCIs, the Frame Relay address has been extended from the original 2-byte address to 3- or 4-byte addresses. EA field defines the number of bytes

Extended Address: Three Address Formats



a. Two-byte address (10-bit DLCI)

b. Three-byte address (16-bit DLCI)

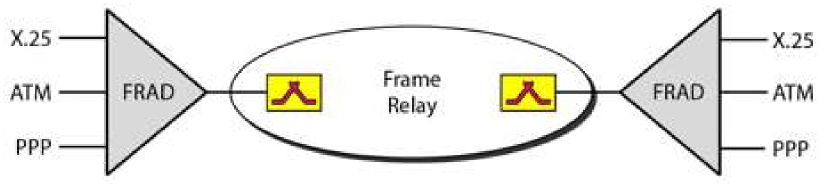
c. Four-byte address (23-bit DLCI)

FRADs

To handle frames arriving from other protocols, Frame Relay uses a device called a Frame Relay assembler/disassembler (FRAD). A FRAD assembles and disassembles frames coming from other protocols to allow them to be carried by Frame Relay frames.

A FRAD can be implemented as a separate device or as part of a switch.

FRAD



VOFR

Frame Relay networks offer an option called Voice Over Frame Relay (VOFR) that sends voice through the network. Voice is digitized using PCM and then compressed. The result is sent as data frames over the network. This feature allows the inexpensive sending of voice over long distances.

However, note that the quality of voice is not as good as voice over a circuit-switched network such as the telephone network. Also, the varying delay mentioned earlier sometimes corrupts real-time voice.

LMI

Frame Relay was originally designed to provide PVC connections. There was not, therefore, a provision for controlling or managing interfaces. Local Management Information (LMI) is a protocol added recently to the Frame Relay protocol to provide more management features. In particular, LMI can provide.

- A keep-alive mechanism to check if data are flowing.
- A multicast mechanism to allow a local end system to send frames to more than one remote end system.
- A mechanism to allow an end system to check the status of a.

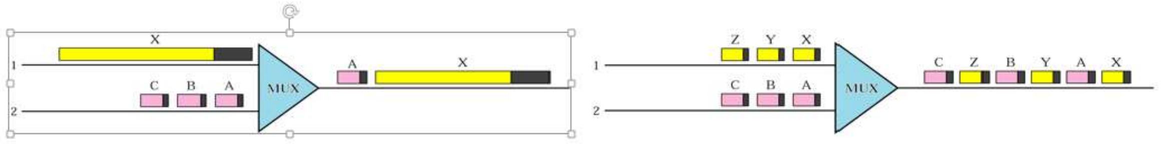
16. ATM

Asynchronous Transfer Mode. ATM is the cell relay protocol designed by ATM forum and adopted by ITU-T. ATM uses asynchronous TDM Cells are transmitted along virtual circuits.

Design Goals:

- Large bandwidth and less susceptible to noise degradation
- Interface with existing systems without lowering their effectiveness
- Inexpensive implementation
- Support the existing telecommunications hierarchies
- Connection-oriented to ensure accurate and predictable delivery
- Many functions are hardware implementable

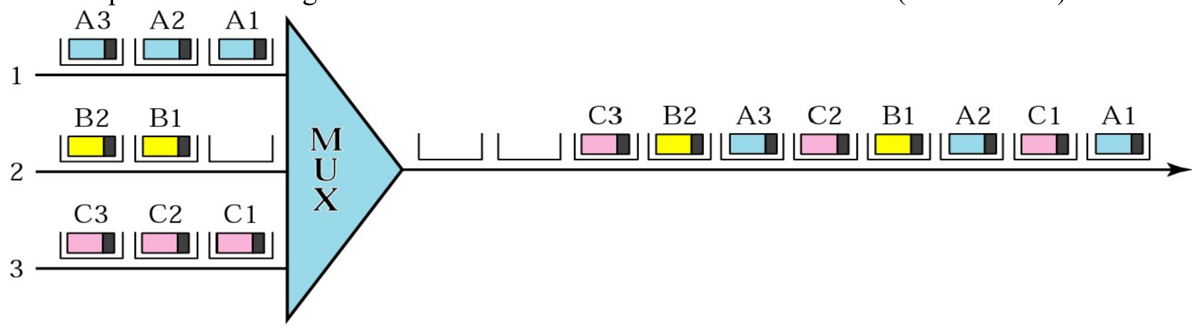
Multiplexing using Cells



- The variety of packet sizes makes traffic unpredictable
- A cell network uses the cell as the basic unit of data exchange
  - A cell is defined as ***a small, fixed sized*** block of information
  - Cells are interleaved so that non suffers a long delay
  - A cell network can handle real-time transmissions
  - Network operation is more efficient and cheaper.

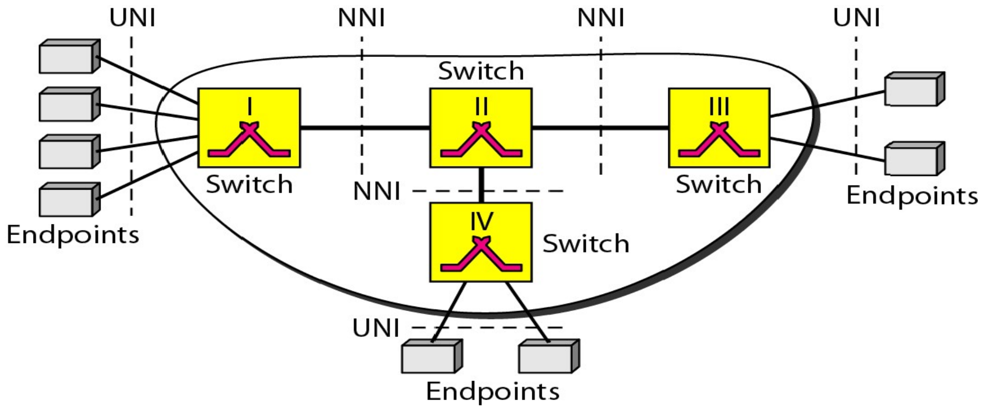
Asynchronous TDM

ATM uses asynchronous time-division multiplexing-that is why it is called Asynchronous Transfer Mode-to multiplex cells corning from different channels. It uses fixed-size slots (size of a cell).



ATM Architecture

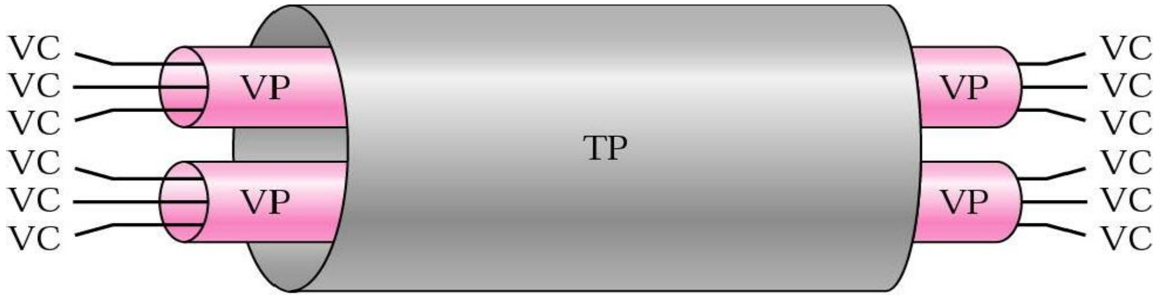
ATM is a cell-switched network. The user access devices, called the endpoints, are connected through a user-to-network interface (UNI) to the switches inside the network. The switches are connected through network-to-network interfaces (NNIs).



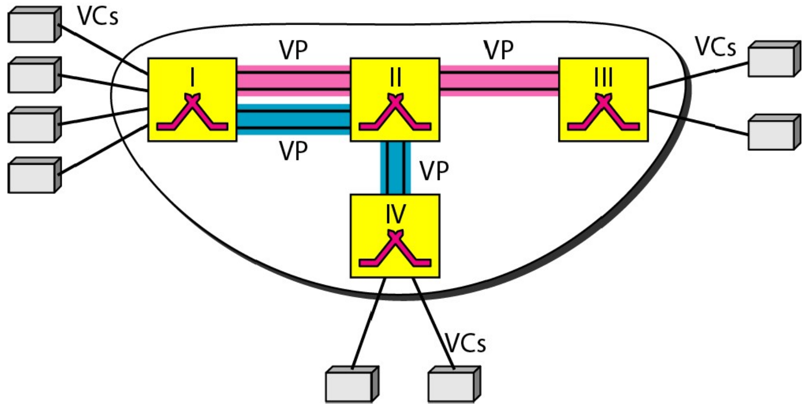
Virtual Connection

- Connection between two endpoints is accomplished through
  - Transmission path (TP)
  - Virtual path (VP)
  - Virtual circuit (VC)
- A virtual connection is defined by a pair of numbers: ***VPI*** and ***VCI***

A transmission path is divided into several virtual paths. A virtual path (VP) provides a connection or a set of connections between two switches. Think of a virtual path as a highway that connects two cities. Each highway is a virtual path; the set of all highways is the transmission path.



As this above figure eight endpoints are communicating using four VCs. However, the first two VCs seem to share the same virtual path from switch I to switch III, so it is reasonable to bundle these two VCs together to form one VP. On the other hand, it is clear that the other two VCs share the same path from switch I to switch IV, so it is also reasonable to combine them to form one VP.



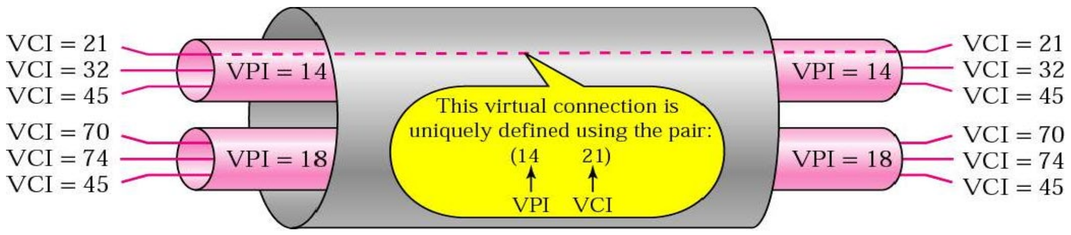
Identifiers and Cells

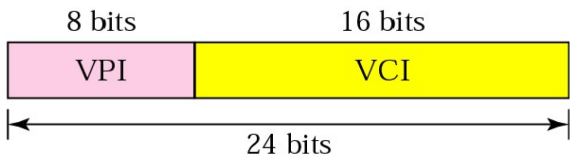
Identifiers In a virtual circuit network, to route data from one endpoint to another, the virtual connections need to be identified. For this purpose, the designers of ATM created a hierarchical identifier with two levels:

Virtual path identifier (VPI)

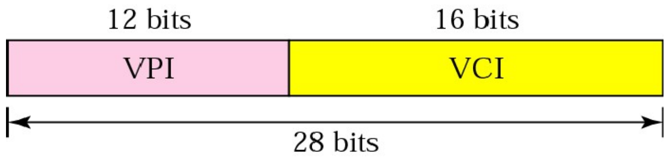
Virtual-circuit identifier (VCl)

The basic data unit in an ATM network is called a cell. A cell is only 53 bytes long with 5 bytes allocated to the header and 48 bytes carrying the payload.



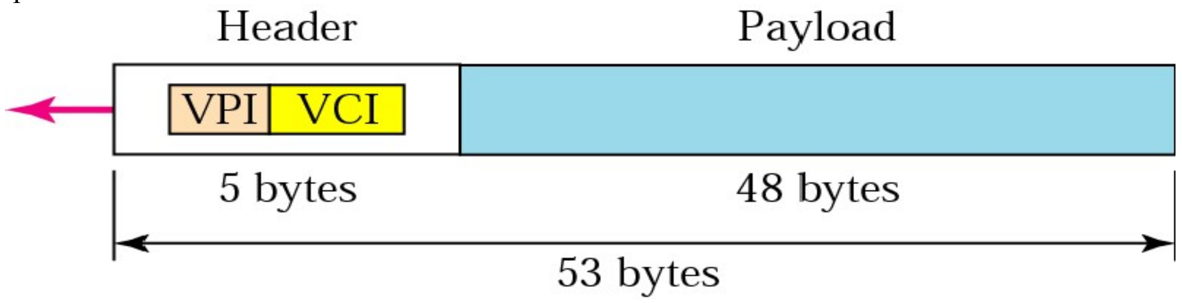


a. VPI and VCI in a UNI



b. VPI and VCI in an NNI

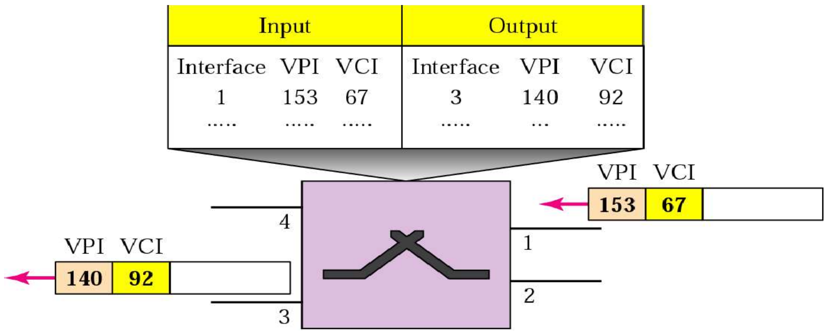
We will study in detail the fields of a cell, but for the moment it suffices to say that most of the header is occupied by the VPI and VCI that define the virtual connection through which a cell should travel from an endpoint to a switch or from a switch to another switch.



Switching

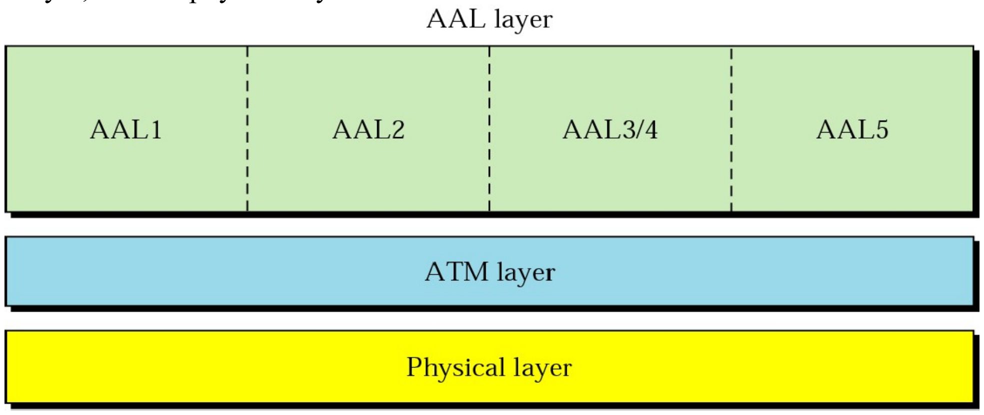
ATM uses switches to route the cell from a source endpoint to the destination endpoint. A switch routes the cell using both the VPls and the VCls. The routing requires the whole identifier.

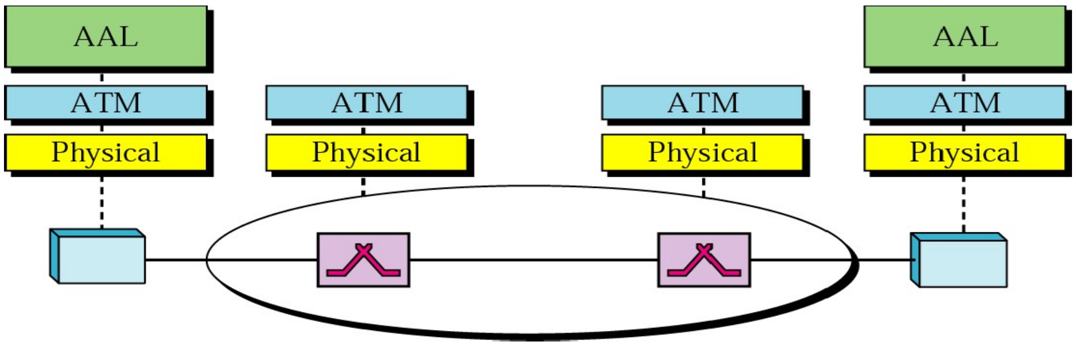
ATM uses switches to route the cell from a source endpoint to the destination endpoint. A switch routes the cell using both the VPls and the VCls. The routing requires the whole identifier.



ATM Layers

The ATM standard defines three layers. They are, from top to bottom, the application adaptation layer, the ATM layer, and the physical layer.

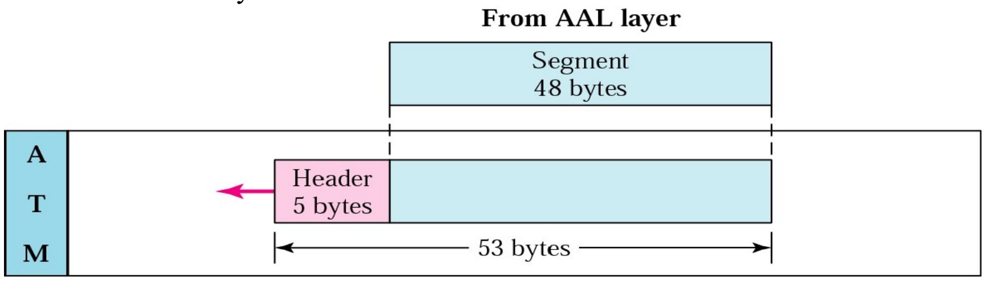




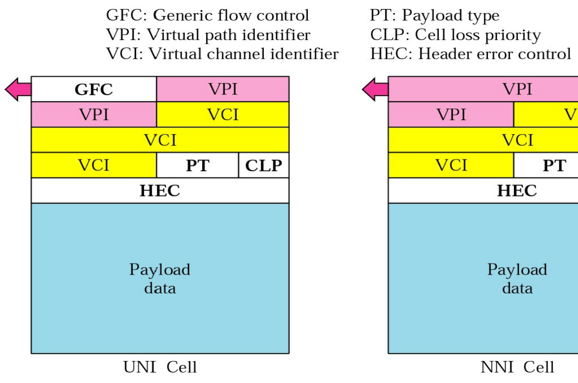
SONET The original design of ATM was based on SONET as the physical layer carrier. SONET is preferred for two reasons. First, the high data rate of SONET's carrier reflects the design and philosophy of ATM. Second, in using SONET, the boundaries of cells can be clearly defined.

ATM Layer and Headers

The ATM layer provides routing, traffic management, switching, and multiplexing services. It processes outgoing traffic by accepting 48-byte segments from the AAL sub layers and transforming them into 53-byte cells by the addition of a 5-byte header

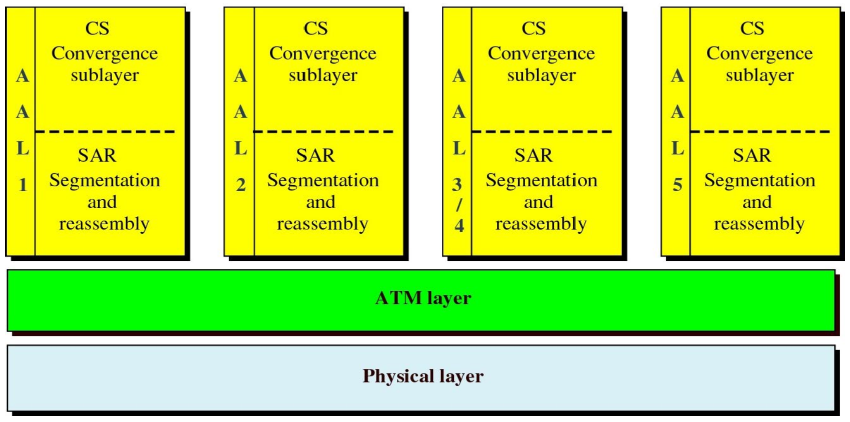


Header Format ATM uses two fonnats for this header, one for user-to-network interface (UNI) cells and another for network-to-network interface (NNI) cells.

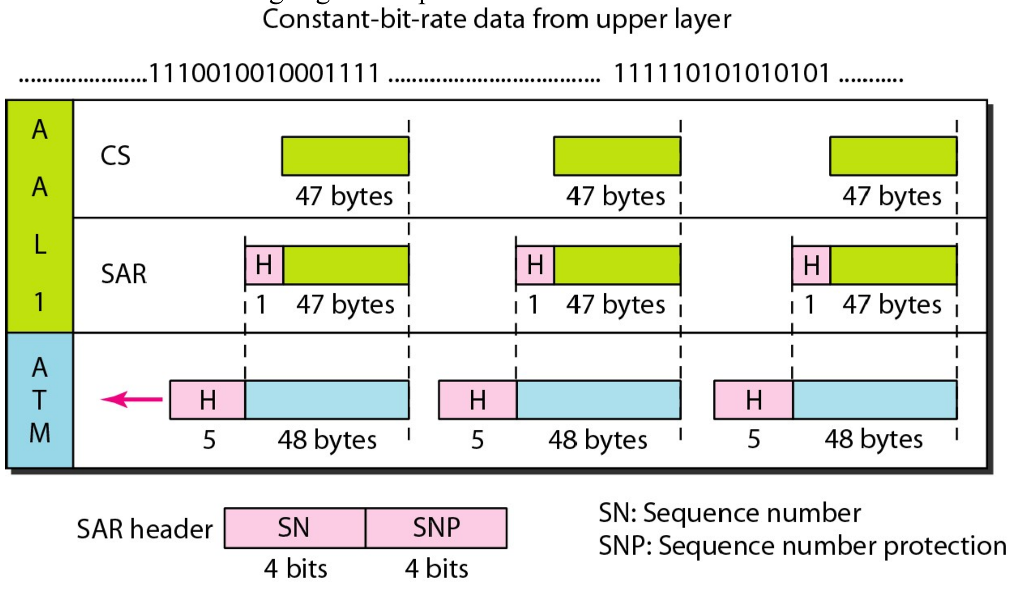


Application Adaptation Layer (AAL)

- Convert data from upper-layer into 48-byte data units for the ATM cells
- AAL1 – constant bit rate (CBR) video and voice
- AAL2 – variable bit rate (VBR) stream ☺low-bit-rate traffic an short-frame traffic such as audio (ex: mobile phone)
- AAL3/4 – connection-oriented/connectionless data.
- AAL5 – SEAL (Simple and Efficient Adaptation Layer)
  No sequencing and error control mechanisms

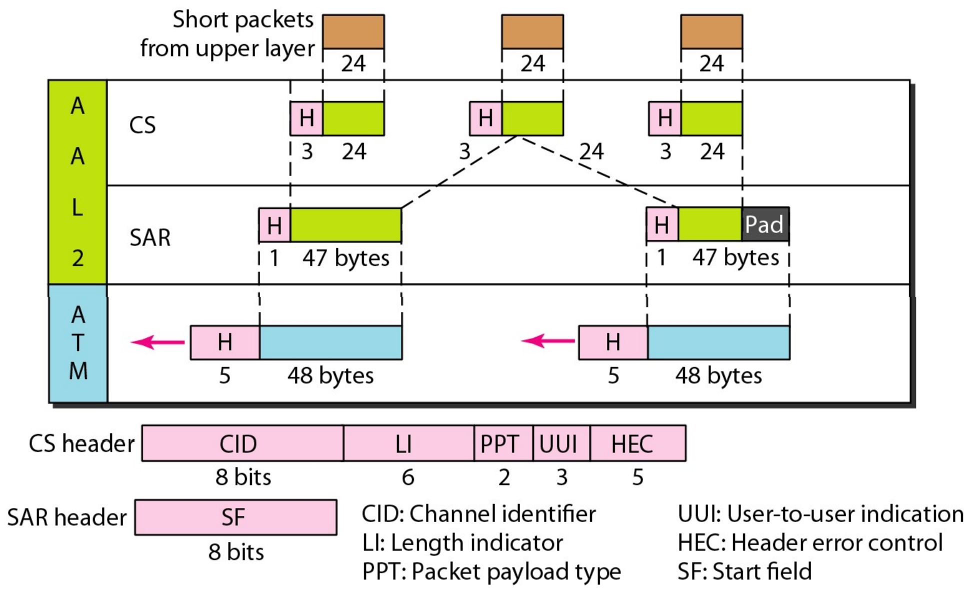


<u>AALI:-</u> AALI supports applications that transfer information at constant bit rates, such as video and voice. It allows ATM to connect existing digital telephone networks such as voice channels and T lines.



Above figure shows the process of encapsulating a short frame from the same source (the same user of a mobile phone) or from several sources (several users of mobile telephones) into one cell).

<u>AAL2</u>.

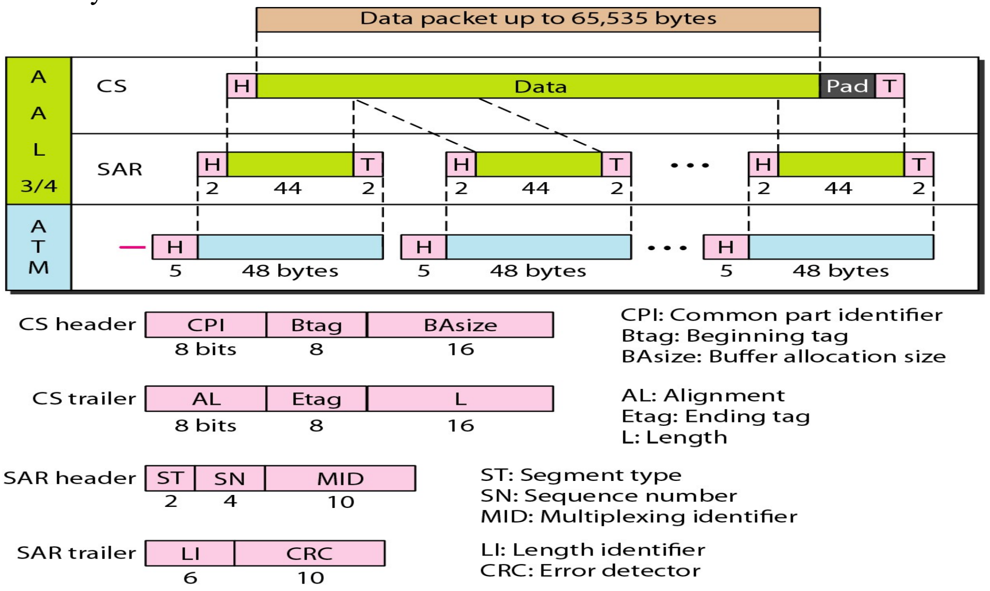


The CS layer overhead consists of five fields:

- Channel identifier (CID). The 8-bit CID field defines the channel (user) of the short packet.
- Length indicator (LI). The 6-bit LI field indicates how much of the final packet is data.
- Packet payload type (PPT). The PPT field defines the type of packet.
- User-to-user indicator (UUI). The UUI field can be used by end-to-end users.
- Header error control (HEC). The last 5 bits is used to correct errors in the header.
- The only overhead at the SAR layer is the start field (SF) that defines the offset from the beginning of the packet.

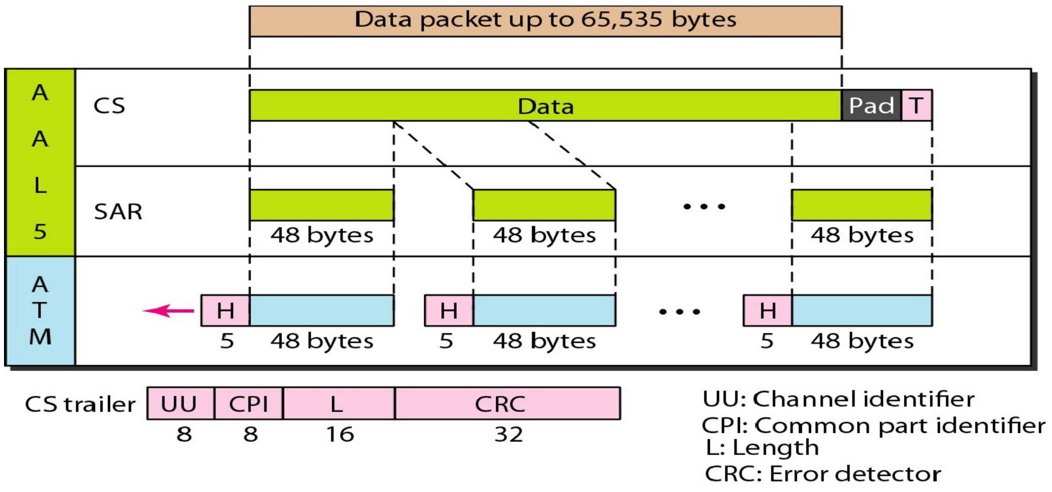
<u>AAL 3/4</u>

AAL3/4 Initially, AAL3 was intended to support connection-oriented data services and AAL4 to support connectionless services. As they evolved, however, it became evident that the fundamental issues of the two protocols were the same. They have therefore been combined into a single format calledAAL3/4. Figure 18.22 shows the AAL3/4 sub layer.


Data packet up to 65,535 bytes
A A L 3/4
CS
H
Data
Pad
T
SAR
H 2 44 2 T
• • •
A T M
H 5 48 bytes
CS header CPI 8 bits Btag 8 BAsize 16
CS trailer AL 8 bits Etag 8 L 16
SAR header ST 2 SN 4 MID 10
SAR trailer LI 6 CRC 10
CPI: Common part identifier
Btag: Beginning tag
BAsize: Buffer allocation size
AL: Alignment
Etag: Ending tag
L: Length
ST: Segment type
SN: Sequence number
MID: Multiplexing identifier
LI: Length identifier
CRC: Error detector

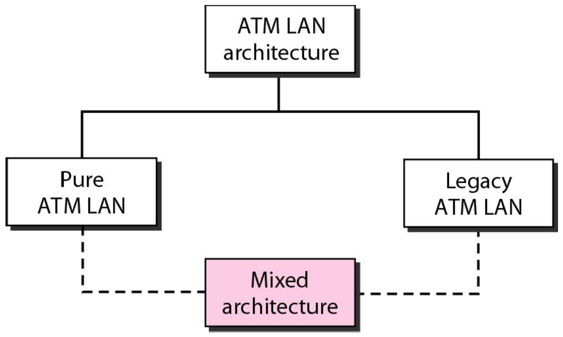
AALS

AALS AAL3/4 provides comprehensive sequencing and error control mechanisms that are not necessary for every application. For these applications, the designers of ATM have provided a fifth AAL sub layer, called the simple and efficient adaptation layer (SEAL). AALS assumes that all cells belonging to a single message travel sequentially and that control functions are included in the upper layers of the sending application.



ATM LAN

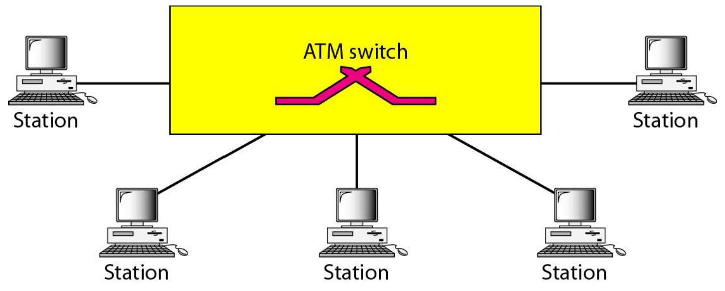
ATM is mainly a wide-area network (WAN ATM), the technology can be adapted to local-area networks (ATM LANs). The high data rate of the technology has attracted the attention of designers who are looking for greater and greater speeds in LANs.



Pure ATM Architecture

In a pure ATM LAN, an ATM switch is used to connect the stations in a LAN, in exactly the same way stations are connected to an Ethernet switch. In this way, stations can exchange data at one of two standard rates of ATM technology (155 and 652 Mbps). However, the station uses a virtual path identifier (VPI) and a virtual circuit identifier (VEl), instead of a source and destination address.

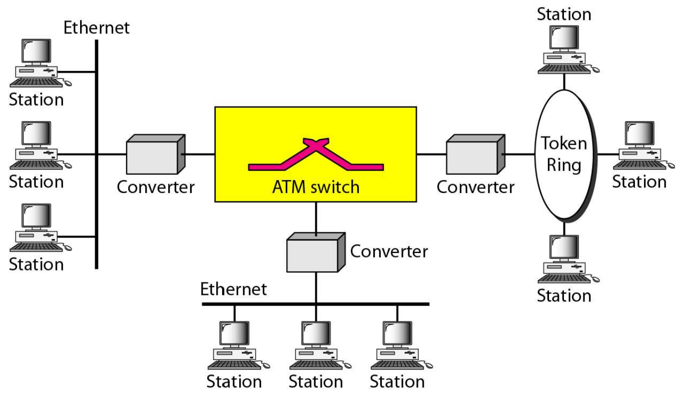
This approach has a major drawback. The system needs to be built from the ground up existing LANs cannot be upgraded into pure ATM LANs.



Legacy LAN Architecture

A second approach is to use ATM technology as a backbone to connect traditional LANs. below figure shows this architecture, a legacy ATM LAN. Stations on the same LAN can exchange data at the rate and format of traditional LANs (Ethernet, Token Ring, etc.). But when two stations on two different LANs need to exchange data, they can go through a converting device that changes the frame format.

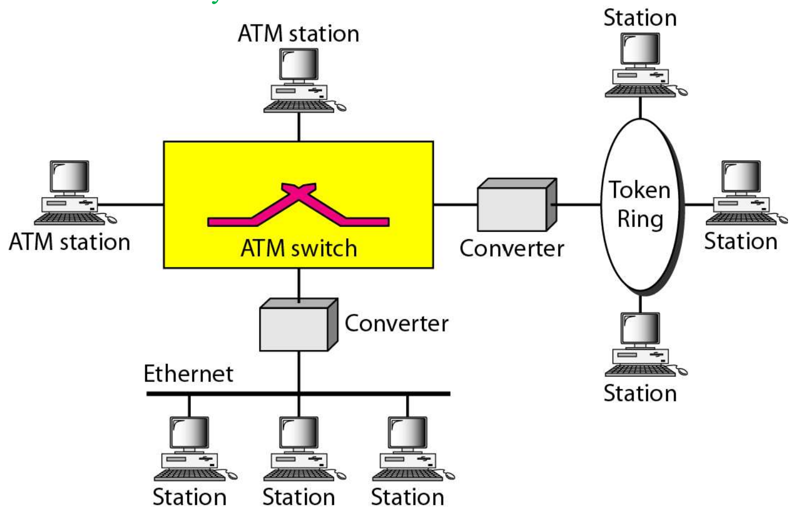
The advantage here is that output from several LANs can be multiplexed together to create a high-data-rate input to the ATM switch.



Mixed Architecture ATM LAN

Probably the best solution is to mix the two previous architectures. This means keeping the existing LANs and, at the same time, allowing new stations to be directly connected to an ATM switch.

The mixed architecture LAN allows the gradual migration of legacy LANs onto ATM LANs by adding more and more directly connected stations to the Switch.

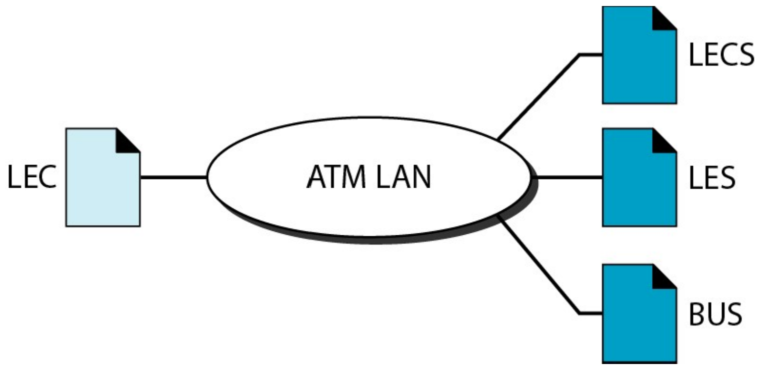


LAN Emulation (LANE)

- Connectionless versus connection-oriented
- Physical addresses versus virtual-circuit identifiers
- Multicasting and broadcasting delivery
- Interoperability
- Client/Server model in a LANE
  - LANE Configuration Server (LECS), LANE Server (LES), LANE Client (LEC)
  - Broadcast/Unknown Server (BUS).

Client/Server Model

LANE is designed as a client/server model to handle the four previously discussed problems. The protocol uses one type of client and three types of servers.



LAN Emulation Client

All ATM stations have LAN emulation client (LEC) software installed on top of the three ATM protocols. These protocols send their requests to LEC for a LAN service such as Connectionless delivery using MAC unicast, multicast, or broadcast addresses.

LAN Emulation Configuration Server

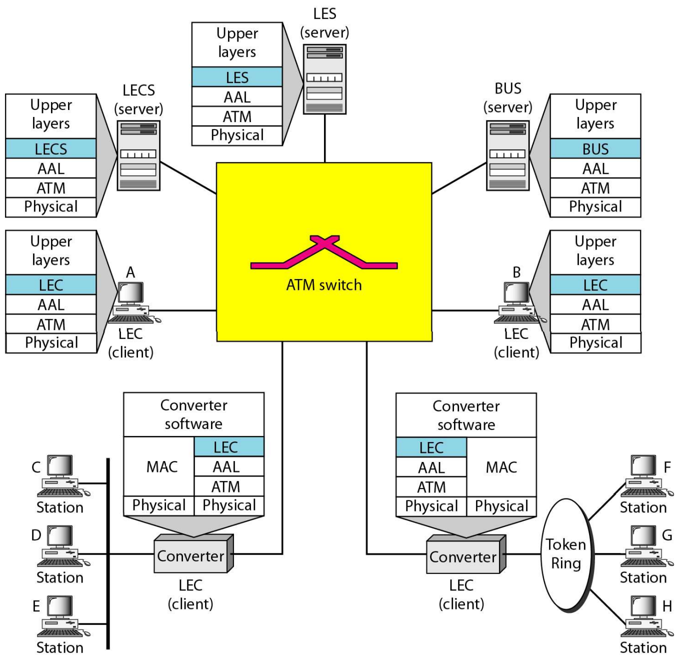
The LAN emulation configuration server (LECS) is used for the initial connection between the client and LANE. This server is always waiting to receive the initial contact.

LAN Emulation Server

LAN emulation server (LES) software is installed on the LES. When a station receives a frame to be sent to another station using a physical address, LEC sends a special frame to the LES. The server creates a virtual circuit between the source and the destination station.

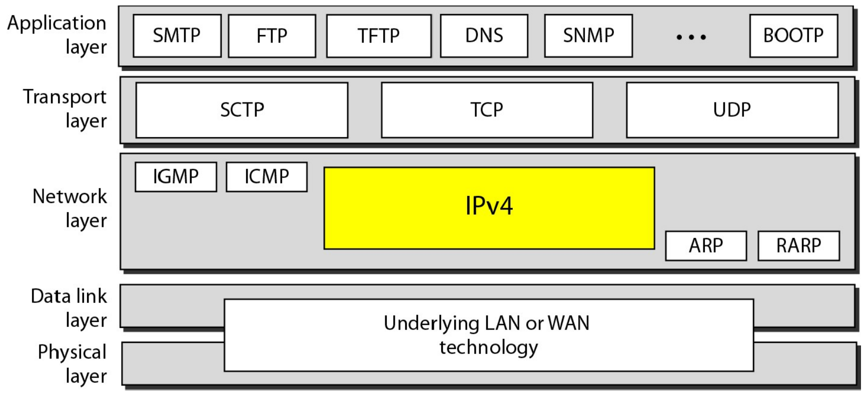
Mixed Architecture Using LANE

Three types of servers are connected to the ATM switch (they can actually be part of the switch). Also we show two types of clients. Stations A and B, designed to send and receive LANE communication, are directly connected to the ATM switch. Stations C, D, E, F, G, and H in traditional legacy LANs are connected to the switch via a converter.



17. Explain about Ipv4 datagram format?

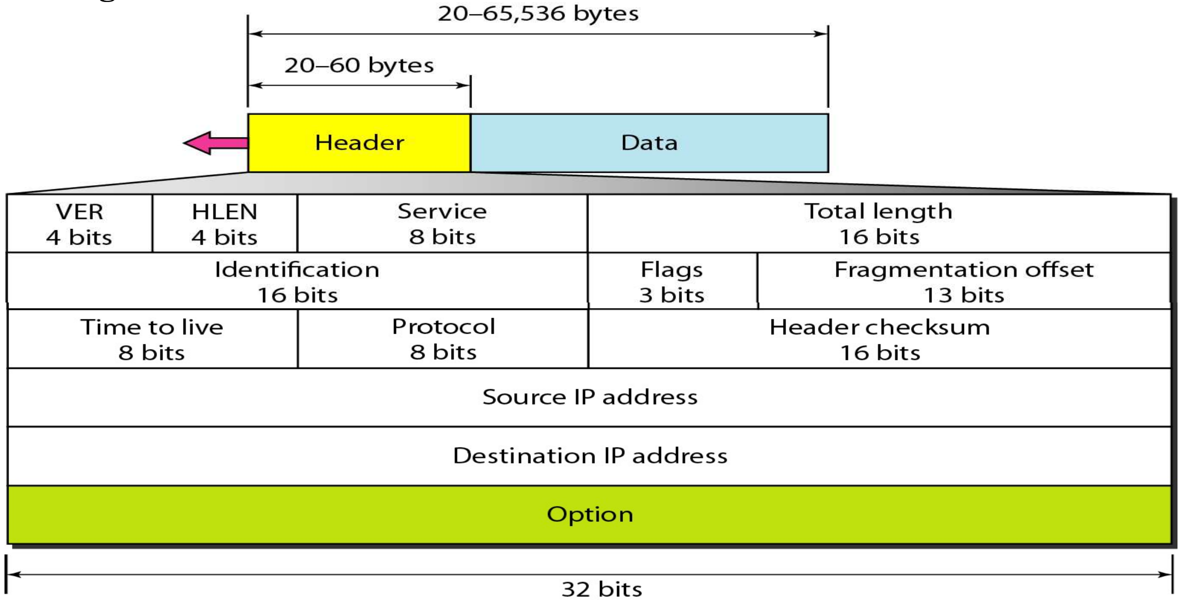
The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/Ip protocols.
IPv4 is an unreliable and connectionless datagram protocol-a best-effort delivery service.
IPv4 is also a connectionless protocol for a packet-switching network that uses the datagram approach.
Position of ipv4 in TCP/IP protocol:



Packets in the IPV4 layer is known as datagram.

Datagram consists of
----Header
----Data

IPV4 datagram format:



The header is 20 to 60 bytes in length and contains information essential to routing and delivery.
It is customary in *TCP/IP* to show the header in 4-byte sections.

VERSION: This 4-bit field defines the version of the IPv4 protocol.

Header length (HLEN):This 4-bit field defines the total length of the datagram header in 4-byte words.
Services:
This is a 8 bit field.

Service type:
In this interpretation, the first 3 bits are called precedence bits. The next 4 bits are called type of service (TOS) bits, and the last bit is not used.
Precedence:

The precedence defines the priority of the datagram in issues such as congestion.

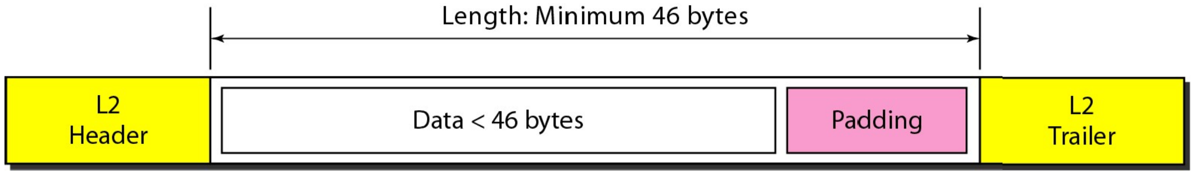
Tos:

TOS bits is a 4-bit subfield with each bit having a special meaning.

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

Total length:

This is a In-bit field that defines the total length (header plus data) of the IPv4 datagram in bytes.



Identification:

A source node gives a unique ID to each packet. Identification, Flags, Fragmentation offset fields are used for fragmentation.

Time to Live (TTL):

A packet has a limited lifetime in the network to avoid zombie packets. Designed to hold a timestamp, and decreased by each router. A packet is discarded by a router if TTL is zero.

It is revised to hold the maximum number of hops the packet can travel thru the network. Each router decrements it by one. It is needed because routing table may be corrupted. Limits the journey of the packet to local network.

Protocol:
This 8-bit field defines the higher-level protocol that uses the services of the IPv4 layer.
An IPv4 datagram can encapsulate data from several higher-level
protocols such as TCP, UDP, ICMP, and IGMP. This field specifies the final destination protocol to which the IPv4 datagram is delivered.
Fragmentation:

Fragmentation is needed because there is a limit on the no. of bits a packet should contain based on MTU.

Maximum transfer unit:

When a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network.
Mtu for some protocols:

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

The source usually does not fragment the IPv4 packet. The transport layer will instead segment the data into a size that can be accommodated by IPv4 and the data link layer in use.

When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but with some changed.
A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU.

In IPv4, a datagram can be fragmented by the source host or any router in the path although there is a tendency to limit fragmentation only at the source. The reassembly of the datagram, however, is done only by the destination host because each fragment becomes an independent datagram.

The final destination host can reassemble the original datagram from the fragments received (if none of them is lost) by using the following strategy:
1. The first fragment has an offset field value of zero.
2. Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.
3. Divide the total length of the first and second fragments by 8. The third fragment
has an offset value equal to that result.
4. Continue the process. The last fragment has a *more* bit value of O.

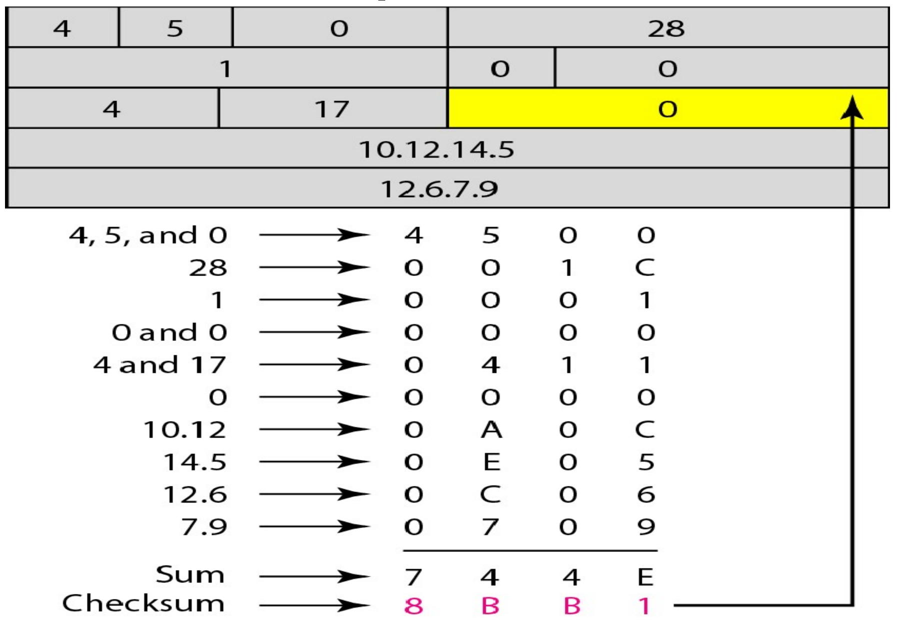
Fields related to fragmentation:

Identification:
This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host.

Flags:
This is a 3-bit field. The first bit is reserved.
The second bit is called the do not fragment bit.
If its value is 1*, the machine must not fragment the datagram. If it* cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host.
If its value is 0, the datagram can be fragmented if necessary. The third bit is called the more fragment bit.
If its value is 1*, it means the datagram is not the last fragment;* there are more fragments after this one.
If its value is 0, it means this is the last or only fragment.
Fragmentation offset:
This 13-bit field shows the relative position of this fragment with respect to the whole datagram.
It is the offset of the data in the original datagram measured in units of 8 bytes.
Remember that the value of the offset is measured in units of 8 bytes.
This forces hosts or routers that fragment data grams to choose a fragment size so that the first byte number is divisible by 8
Checksum:
Checksum only covers the header and not data.

Data checksum is handled by higher-level protocols that encapsulate the data in the IP datagram.Header changes when packets travels on the network but data does not change.

Calculating Checksum:Divide the IP header into 16-bit sections. Value of checksum field is set to zero. All the sections are added and the sum is complemented. The result is inserted in the checksum field.



Source address:This 32-bit field defines the IPv4 address of the source. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

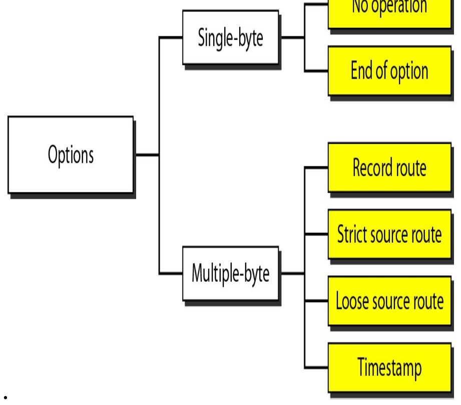
Destination address:

This 32-bit field defines the IPv4 address of the destination.

This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host

Options:

The header of the IPv4 datagram is made of two parts:
a fixed part and a variable part.
The fixed part is 20 bytes long and was discussed in the previous section.
The variable part comprises the options that can be a maximum of 40 bytes.



.
They can be used for network testing and debugging

18. How to improve Quality of Service. Explain Integrated Services and Differentiated Services?

A) **QUALITY OF SERVICE:**

Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.

TECHNIQUES TO IMPROVE QoS:

The four common methods to improve the Quality of Service:

- Scheduling
- Traffic shaping
- Admission control
- Resource reservation.

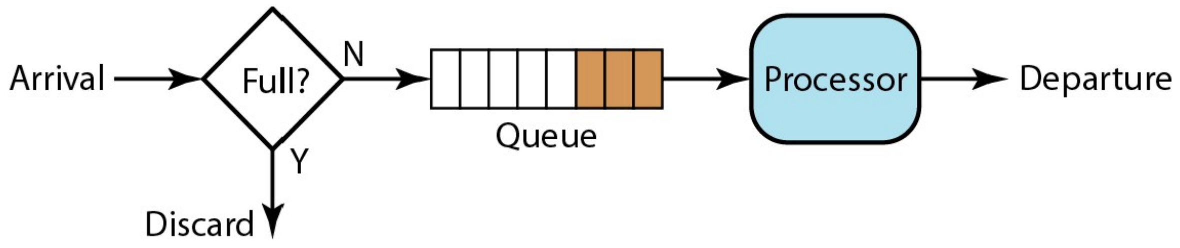
SCHEDULING:

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:

1. FIFO queuing
2. Priority queuing and
3. Weighted fair queuing.

FIFO Queuing:

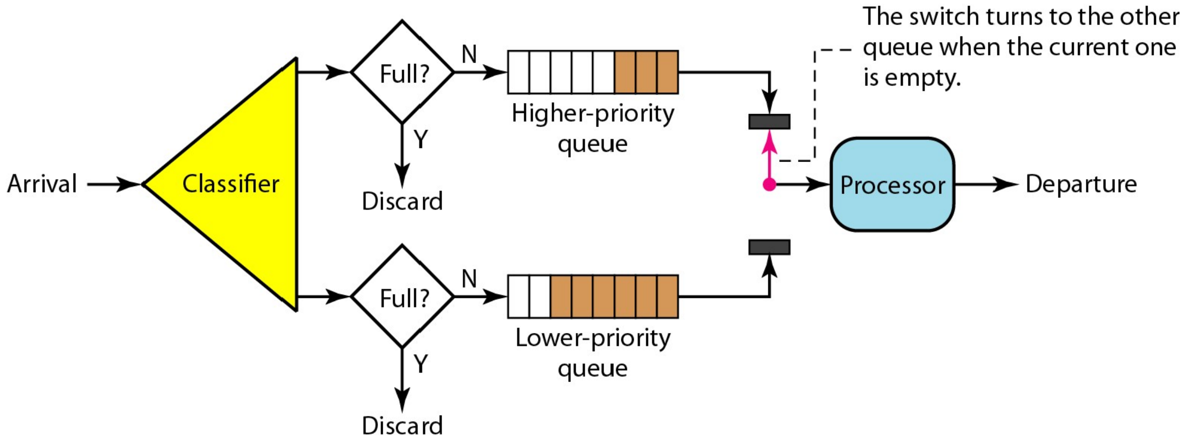
In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop.



Conceptual view of FIFO queue

Priority queue:

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

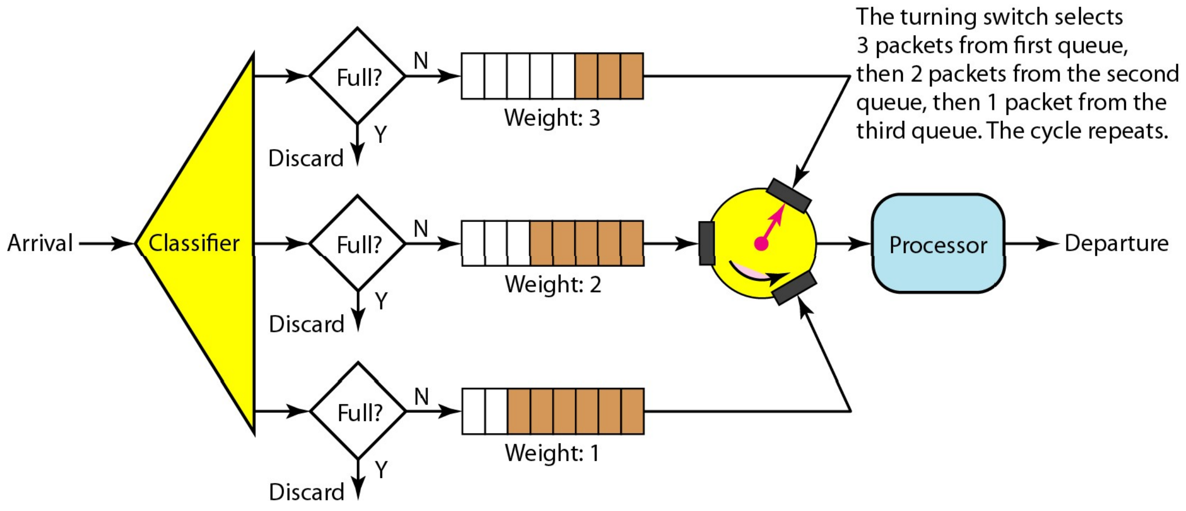


Priority queuing

does not stop serving a queue until it is empty. A priority queue can provide better QoS than the FIFO queue because higher- priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called starvation.

Weighted Fair Queuing:

A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, how- ever, are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight. For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority.

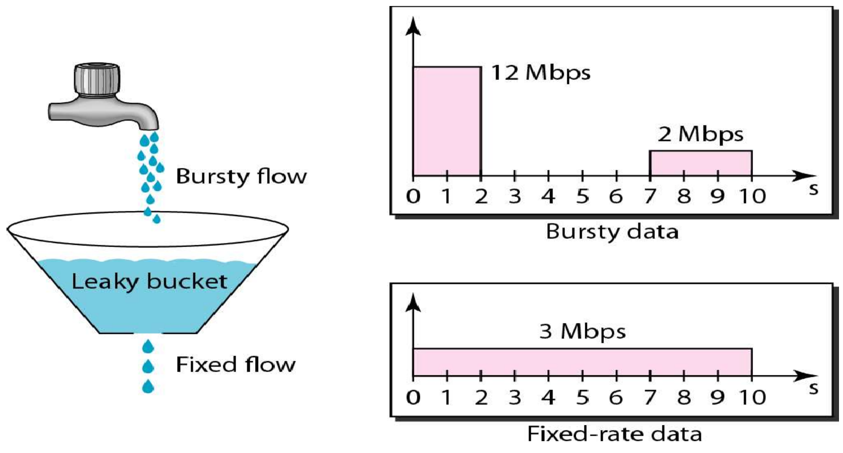

The turning switch selects 3 packets from first queue, then 2 packets from the second queue, then 1 packet from the third queue. The cycle repeats.
Arrival
Classifier
Full?
N
Y
Discard
Weight: 3
Full?
N
Y
Discard
Weight: 2
Full?
N
Y
Discard
Weight: 1
Processor
Departure

Weighted fair queuing

TRAFFIC SHAPING:

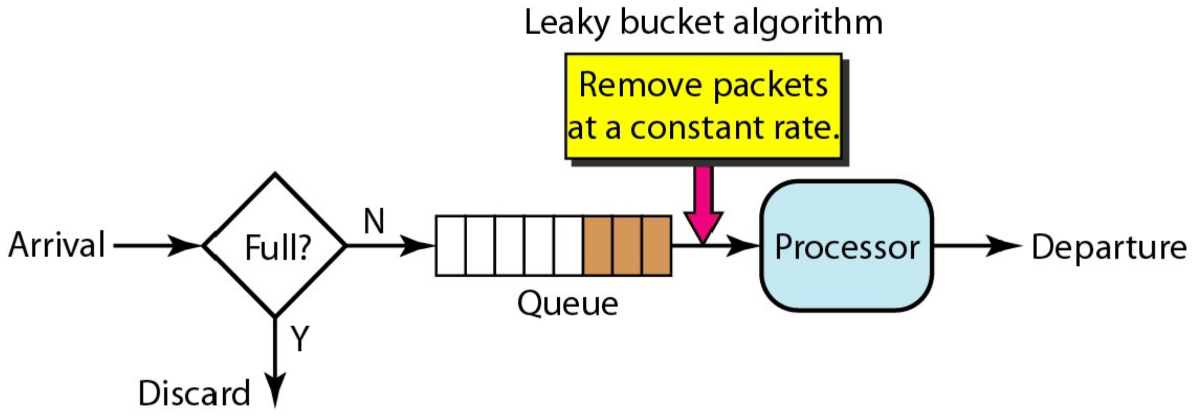
Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic:

- 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 a 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 burst traffic. Bursty chunks are stored in the bucket and sent out at an average rate. In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure, the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10s. The leaky bucket smooths the traffic by sending out data at a rate of3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion. As an analogy, consider the freeway during rush hour (bursty traffic). If, instead, commuters could stagger their working hours, congestion on our freeways could be avoided.



A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits. The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.

2. If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until n is smaller than the packet size.

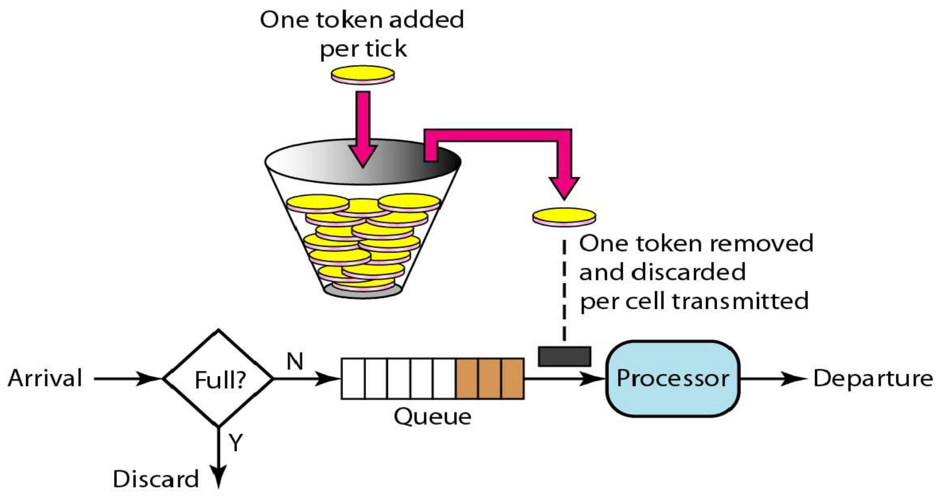
3. Reset the counter and go to step1.

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Token Bucket:

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends n tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

The token bucket allows bursty traffic at a regulated maximum rate.



Token bucket

Combining Token Bucket and Leaky Bucket:

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

RESOURCE RESERVATION:

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand. We discuss in this section one QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

ADMISSIONCONTROL:

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

Based on characteristics and techniques of Quality of Service two models have been designed to provide quality of service in the Internet:

- Integrated Services and
- Differentiated Services.

Both models emphasize the use of quality of service at the network layer (IP), although the model can also be used in other layers such as the data link. IP was originally designed for best-effort delivery. This means that every user receives the same level of services. This type of delivery does not guarantee the minimum of a service, such as bandwidth, to applications such as real-time audio and video. If such an application accidentally gets extra bandwidth, it may be detrimental to other applications, resulting in congestion.

INTEGRATED SERVICES:

Integrated Services, sometimes called IntServ, is a flow-based QoS model, which means that a user needs to create a flow, a kind of virtual circuit, from the source to the destination and inform all routers of the resource requirement.

Integrated Services is a flow-based QoS model designed for IP.

Signaling:

To implement a flow-based model over a connectionless protocol signaling is used. Signaling protocol is to run over IP that provides the signaling mechanism for making a reservation. This protocol is called Resource Reservation Protocol (RSVP).

Flow Specification:

When a source makes a reservation, it needs to define a flow specification. A flow specification has two parts: Rspec(resource specification)and Tspec(traffic specification). Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.). Tspec defines the traffic characterization of the flow.

Admission:

After a router receives the flow specification from an application, it decides to admit or deny the service. The decision is based on the previous commitments of the router and the current availability of the resource.

Service Classes:

Two classes of services have been defined for Integrated Services: guaranteed service and controlled-load service.

Guaranteed Service Class:

This type of service is designed for real-time traffic that needs a guaranteed minimum end-to-end delay. The end-to-end delay is the sum of the delays in the routers, the propagation delay in the media, and the setup mechanism. Only the first, the sum of the delays in the routers, can be guaranteed by the router. This type of service guarantees that the packets will arrive within a certain delivery time and are not discarded if flow traffic stays within the boundary of Tspec. We can say that guaranteed services are quantitative services, in which the amount of end-to-end delay and the data rate must be defined by the application.

Controlled-Load Service Class:

This type of service is designed for applications that can accept some delays, but are sensitive to an overloaded network and to the danger of losing packets. Good examples of these types of applications are file transfer, e-mail, and Internet access. The controlled- load service is a qualitative type of service in that the application requests the possibility of low-loss or no-loss packets.

RSVP:

In the Integrated Services model, an application program needs resource reservation. As we learned in the discussion of the IntServ model, the resource reservation is for a flow. This means that if we want to use IntServ at the IP level, we need to create a flow, a kind of virtual-circuit network, out of the IP, which was originally designed as a datagram packet-switched network. A virtual-circuit network needs a signaling system to set up the virtual circuit before data traffic can start. The Resource Reservation Protocol (RSVP) is
a signaling protocol to help IP create a flow and consequently make a resource discussing reservation. Before RSVP, we need to mention that it is an independent protocol separate Services model. from the Integrated It may be used in other models in the future.

Multicast Trees:

RSVP is different from some other signaling systems we have seen before in that it is a signaling system designed for multicasting. However, RSVP can be also used for unicasting because unicasting is just a special case of multicasting with only one member in the multicast group. The reason for this design is to enable RSVP to provide resource reservations for all kinds of traffic including multimedia which often uses multicasting.

Receiver-Based Reservation:

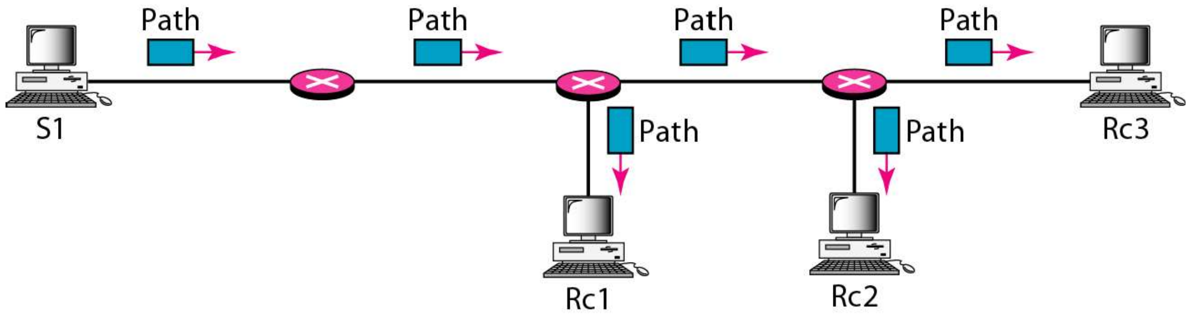
In RSVP, the receivers, not the sender, make the reservation. This strategy matches the other multicasting protocols. For example, in multicast routing protocols, the receivers, not the sender, make a decision to join or leave a multicast group.

RSVP Messages:

RSVP has several types of messages. Two of them: Path and Resv.

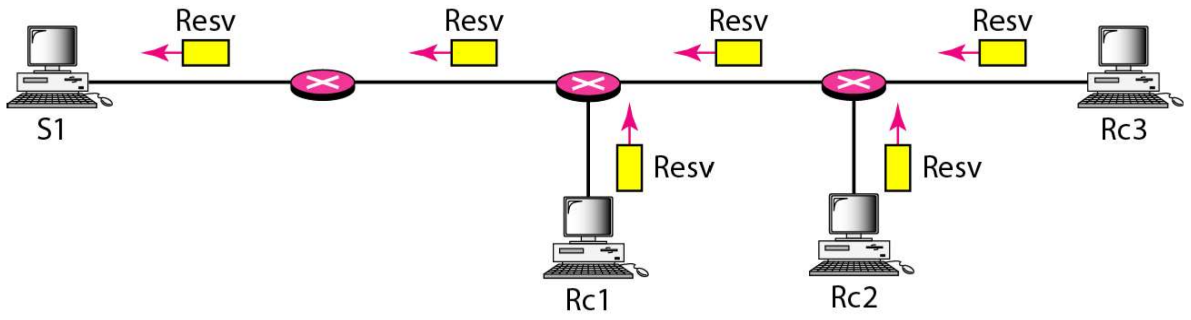
Path Messages :

Recall that the receivers in a flow make the reservation in RSVP. However, the receivers do not know the path traveled by packets before the reservation is made. The path is needed for the reservation. To solve the problem, RSVP uses Path messages. A Path message travels from the sender and reaches all receivers in the multi- cast path. On the way, a Path message stores the necessary information for the receivers. A Path message is sent in a multicast environment; a new message is created when the path diverges.



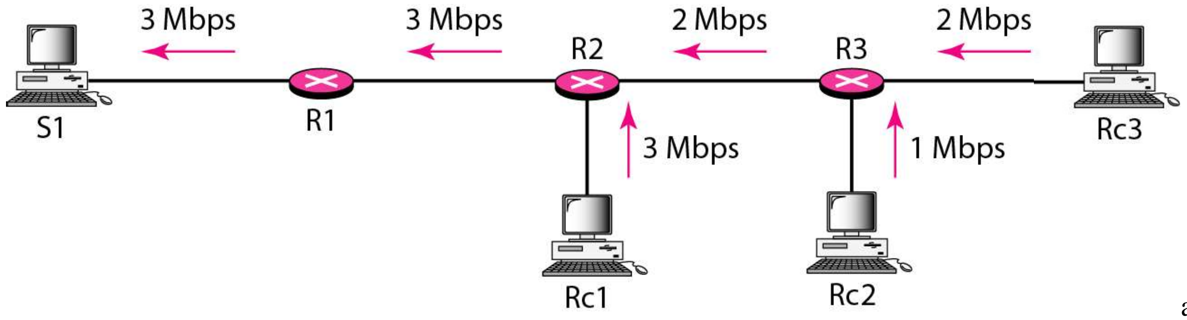
Resv Messages:

After a receiver has received a Path message, it sends a Resv message. The Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP. If a router does not support RSVP on the path, it routes the packet based on the best-effort delivery methods.



Reservation Merging:

In RSVP, the resources are not reserved for each receiver in a flow; the reservation is merged. In Figure, Rc3 requests a 2-Mbps bandwidth while Rc2 requests a I-Mbps bandwidth. Router R3, which needs to make a bandwidth reservation, merges the two requests. The reservation is made for 2 Mbps, the larger of the two, because a 2-Mbps input reservation can handle both requests. The same situation is true for R2. The reader may ask why Rc2 and Rc3, both belonging to one single flow, request different amounts of bandwidth. The answer is that, in a multimedia environment, different receivers may handle different grades of quality. For example, Rc2 may be able' to receive video only at 1 Mbps (lower quality), while Rc3 may be

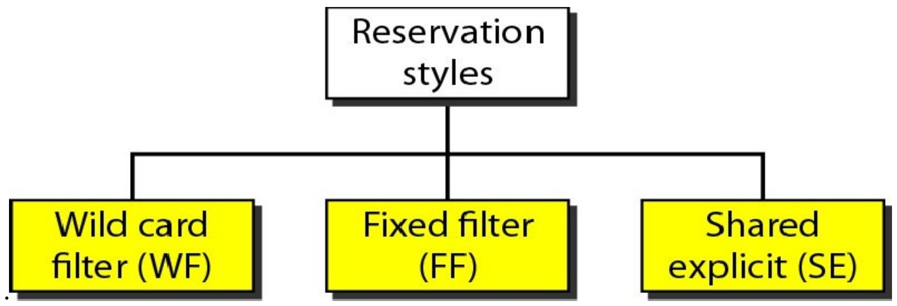


able to

receive video at 2 Mbps (higher quality)

Reservation Styles:

When there is more than one flow, the router needs to make a reservation to accommodate all of them. RSVP defines three types of reservation styles.



Wild Card Filter Style: In this style, the router creates a single reservation for all senders. The reservation is based on the largest request. This type of style is used when the flows from different senders do not occur at the same time.

Fixed Filter Style: In this style, the router creates a distinct reservation for each flow. This means that if there are n flows, n different reservations are made. This type of style is used when there is a high probability that flows from different senders will occur at the same time.

Shared Explicit Style: In this style, the router creates a single reservation which can be shared by a set of flows.

Soft State:

The reservation information (state) stored in every node for a flow needs to be refreshed periodically. This is referred to as a soft state as compared to the hard state used in other virtual-circuit protocols such as ATM or Frame Relay, where the information about the flow is maintained until it is erased. The default interval for refreshing is currently 30 s.

Problems with Integrated Services:

There are at least two problems with Integrated Services that may prevent its full implementation in the Internet: scalability and service-type limitation.

Scalability:

The Integrated Services model requires that each router keep information for each flow. As the Internet is growing every day, this is a serious problem.

Service-Type Limitation:

The Integrated Services model provides only two types of services, guaranteed and control-load. Those opposing this model argue that applications may need more than these two types of services.

DIFFERENTIATED SERVICES:

Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services. Two fundamental changes were made:

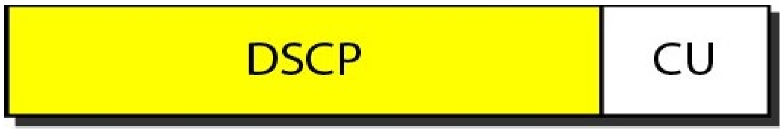
1. The main processing was moved from the core of the network to the edge of the network. This solves the scalability problem. The routers do not have to store information about flows. The applications, or hosts, define the type of service they need each time they send a packet.

2. The per-flow service is changed to per-class service. The router routes the packet based on the class of service defined in the packet, not the flow. This solves the service-type limitation problem. We can define different types of classes based on the needs of applications.

Differentiated Services is a class-based QoS model designed for IP.

DS Field:

In Diffserv, each packet contains a field called the DS field. The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router. IETF proposes to replace the existing TOS (type of service) field in IPv4 or the class field in IPv6 by the DS field.



The DS field contains two subfields: DSCP and CU. The DSCP (Differentiated Services Code Point) is a 6-bit subfield that defines the per-hop behavior (PHB). The 2-bit CU (currently unused) subfield is not currently used. The Diffserv capable node (router) uses the DSCP 6 bits as an index to a table defining the packet-handling mechanism for the current packet being processed.

Per-Hop Behavior:

The Diffserv model defines per-hop behaviors (PHBs) for each node that receives a packet. So far three PHBs are defined: DE PHB, EF PHB, and AF PHB.

DE PHB: The DE PHB (default PHB) is the same as best-effort delivery, which is compatible with TOS.

EF PHB: The EF PHB (expedited forwarding PHB) provides the following services:

- Low loss
- Low latency
- Ensured bandwidth

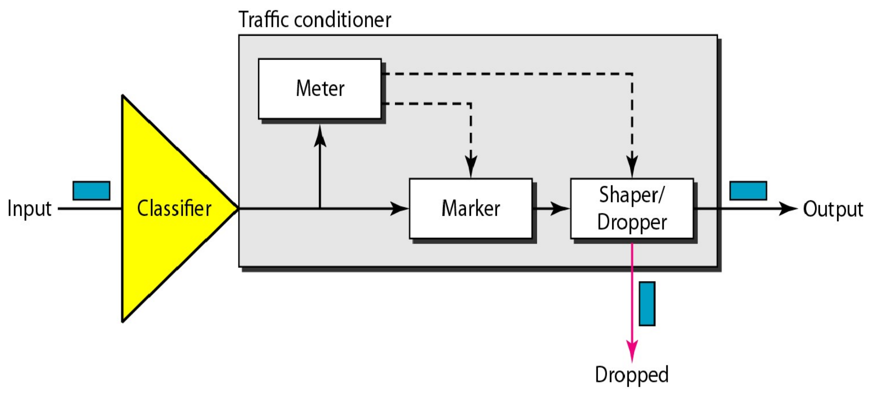
This is the same as having a virtual connection between the source and destination.

AF PHB: The AF PHB (assured forwarding PHB) delivers the packet with a high assurance as long as the class traffic does not exceed the traffic profile of the node. The users of the network need to be aware that some packets may be discarded.

Traffic Conditioner:

To implement Diffserv, the OS node uses traffic conditioners such as meters, markers, shapers, and droppers.

Meters: The meter checks to see if the incoming flow matches the negotiated traffic profile. The meter also sends this result to other components. The meter can use several tools such as a token bucket to check the profile.



Marker: A marker can remark a packet that is using best-effort delivery (OSCP: 000000) or down-mark a packet based on information received from the meter. Down- marking (lowering the class of the flow) occurs if the flow does not match the profile. A marker does not up-mark (promote the class) a packet.

Shaper: A shaper uses the information received from the meter to reshape the traffic if it is not compliant with the negotiated profile.

Dropper: A dropper, which works as a shaper with no buffer, discards packets if the flow severely violates negotiated profile.

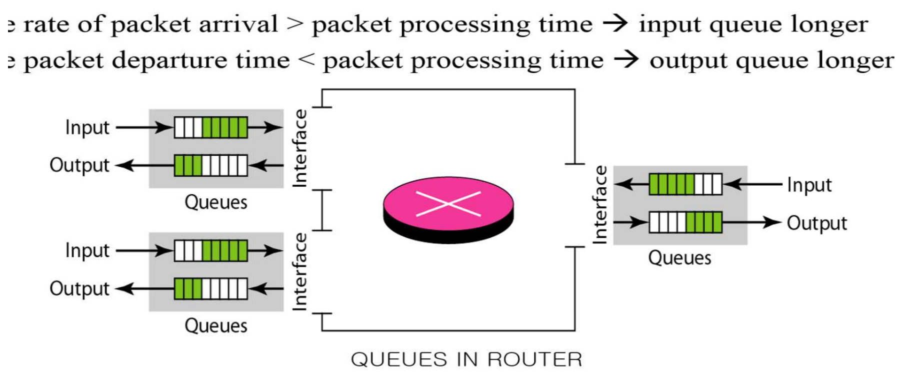
19. **Describe about congestion control?**

CONGESTION:

The load on the network is greater than the capacity of the network.

CONGESTION CONTROL:

- The mechanisms to control the congestion and keep the load below the capacity.
- Congestion occurs because routers and switches have queues – buffers that hold the packets before and after processing.
- The rate of packet arrival > packet processing time -> input queue longer.
- The packet departure time < packet processing time -> output queue longer.


rate of packet arrival > packet processing time → input queue longer

packet departure time < packet processing time → output queue longer

QUEUES IN ROUTER
Computer Networks

CONGESTION CONTROL:

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

OPEN LOOP CONGESTION CONTROL

Retransmission policy:

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.

Window policy:

The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control.

- In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver.
- The Selective Repeat window , on the other hand, tries to send the specific packets that have been lost or corrupted.

Acknowledgement policy:

The acknowledgement policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives ,it may slow down the sender and help prevent congestion.

A receiver may send an acknowledgement only if it has a packet to be sent or a special timer expires. A receiver may decide to acknowledgement only N packets at a time.

Discarding policy:

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.

Admission policy:

An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual- circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network.

CLOSED-LOOP CONGESTION CONTROL

Closed loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols.

Back pressure:

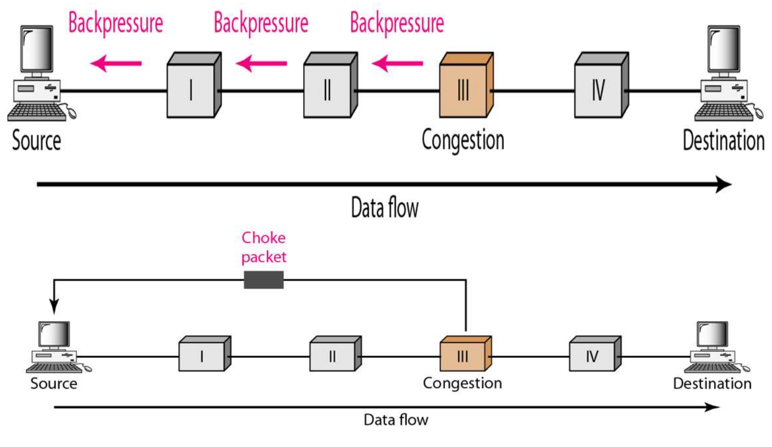
The technique of back pressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.

Back pressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source.

This technique can be applied only to virtual circuit networks.

Choke packet:

A choke packet is a packet sent by a node to the source to inform it of congestion.



Computer Networks

Implicit Signalling:

In implicit signalling ,there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms.

Explicit Signalling:

The node that experiences congestion can explicitly send to the source or destination. This method is different from choke packet method.

we will see this method in Frame Relay congestion control, can occur in either the forward or backward direction.

Backward Signalling:

A bit can set in a packet moving in the direction opposite to the congestion .This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

Forwarding Signalling:

A bit can be set in a packet moving in the direction of the congestion.

This bit can warn the destination that there is congestion. The receiver in this case use policies, such as slowing down the acknowledgements, to alleviate the congestion.

20. **With neat diagrams explain the following:**

a) TELNET

b)FTP

c)EMAIL

A. **TELNET:**

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

TELNET undergoes the following processes:

- Time sharing environment
- Logging
- Network virtual terminal(NVT)
- Embedding
- Options
- Mode of operation

TIME SHARING ENVIRONMENT:

- TELNET was designed at a time when most operating systems, such as UNIX, were operating in a timesharing environment.
- In such an environment, a large computer supports multiple users.
- The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse.
- Even a microcomputer can simulate a terminal with a terminal emulator.

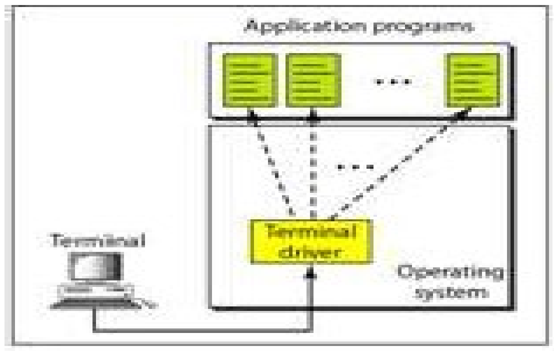
LOGGING:

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

The logging consists of-

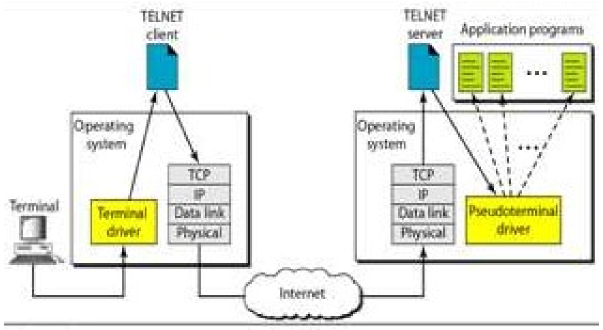
1. Local log-in and
2. Remote log-in

LOCAL LOG-IN:



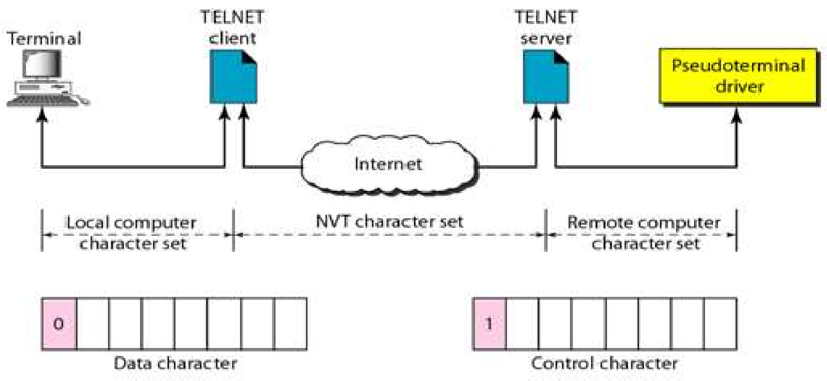
- A user logs into a local time sharing computer using ID and password, so called local log-in.
- The interaction between a user and the computer occurs through a terminal, which is composed of keyboard, monitor, and mouse.

REMOTE LOG-IN:



- The user sends keystrokes to the terminal driver.
- The local OS accepts the characters but not interprets them. It sends these characters to the TELNET client.
- TELNET client transform the characters into a universal character set called network virtual terminal (NVT) characters.
- TELNET server changes NVT to the characters understandable by the remote computer.
- A pseudoterminal driver is added to pass the characters from TELNET server to OS.
- The remote OS interprets and invokes the application.

NETWORK VIRTUAL TERMINAL:

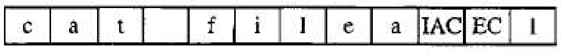


- The mechanism to access a remote computer is complex. This is so because every computer and its operating system accept a special combination of characters as tokens. For example, the end-of-file token in a computer running the DOS operating system is
  Ctrl+z, while the UNIX operating system recognizes Ctrl+d.
- We are dealing with heterogeneous systems.
- If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer.
- TELNET solves this problem by defining a universal interface called the network virtual terminal (NVT) character set.
- Using this interface, TELNET translates data and commands from NVT form into the form acceptable by the remote computer.
- For data, NVT is an 8-bit character set in which the 7 lowest-order bits are the same as ASCII and the highest-order bit is O. To send control characters between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest-order bit is set to l.
- Some of the control characters and their meanings are listed below:

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

EMBEDDING:

- TELNET uses embedding concept to allow one TCP connection to be used for both data and control. (The TELNET server is at well known port 23 and TELNET client is at ephemeral port.)
- Control data is embedded in data stream by a preceding IAC (Interpret As Control) character.
- Example:
  If a user want to type a command “cat file1” on a remote computer but she mistypes it as “cat filea” and uses the backspace key to correct it. (cat=display) In default implementation of TELNET, the user cannot edit command locally, the editing is done at the server. Therefore, the command she has typed is “catfilea<backspace>1”. The backspace is translated into two remote characters, IAC and EC, which are embedded into the data and sent to remote server.



Typed at the remote terminal

OPTIONS:

- Options are extra features for users with more sophisticated terminal.
- TELNET also allows client and server to negotiate options before or during the use of service**.**
- The table shows some common options:

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

OPTION NEGOTIATION: Option Negotiation To use any of the options mentioned in the previous section first requires option negotiation between the client and the server. Four control characters are used for this purpose.

The four control characters are listed below:

NVT character set for option negotiation

Character	*Decimal*	*Binary*	*Meaning*
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

Offer

- To enable an option, A party sends WILL command (WILL I enable the option?), the other party sends either DO (Please DO) or DONT (Please DON'T).
- To disable an option, A party sends WONT command (I WON'T use this option anymore), the other party sends DONT (DON'T use it anymore).

Request

- To enable an option, A party sends DO command (Please DO enable the option), the other party sends either WILL (I WILL) or WONT (I WON'T).
- To disable an option, A party sends DONT command (Please DON'T use this option anymore), the other party sends WONT (I WON'T).

MODES OF OPERATION:

Most TELNET implementations operate in one of three modes:

1. default mode,
2. character mode, or
3. line mode.

DEFAULT MODE:

The default mode is used if no other modes are invoked through option negotiation. In this mode, the echoing is done by the client. The user types a character, and the client echoes the character on the screen (or printer) but does not send it until a whole line is completed.

CHARACTER MODE:

In the character mode, each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed if the transmission time is long (such as in a satellite connection). It also creates overhead (traffic) for the network because three TCP segments must be sent for each character of data.

LINE MODE:

A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode. In this mode, called the line mode, line editing (echoing, character erasing, line erasing, and so on) is done by the client. The client then sends the whole line to the server.

ELECTRONIC MAIL

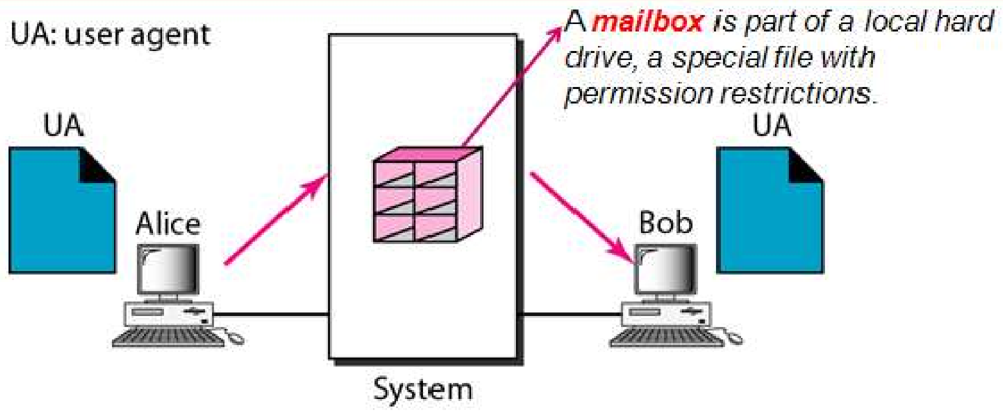
One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program

THE TOPICS UNDER E-MAIL ARE-

- Architecture (UA,MTA,MAA)
- User Agent
- Message Transfer Agent: SMTP
- Message Access Agent: POP and IMAP
- Web-Based Mail

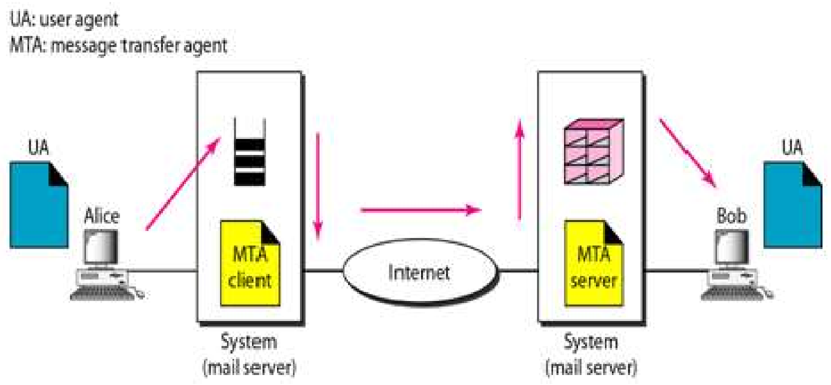
ARCHITECTURE:To explain the architecture of e-mail, we give four scenarios. We begin with the simplest situation and add complexity as we proceed. The fourth scenario is the most common in the exchange of email.

FIRST SCENARIO:


UA: user agent
A mailbox is part of a local hard drive, a special file with permission restrictions.
UA
UA
Alice
Bob
System

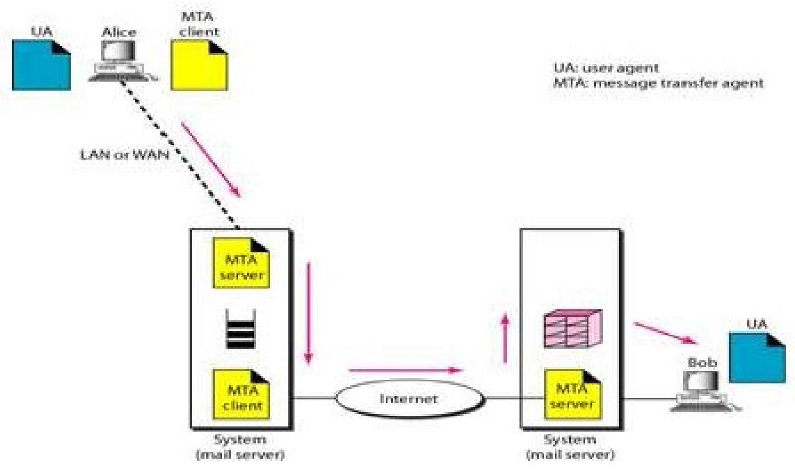
- Alice and Bob are directly connected to the same shared system.
- Alice run a UA program to prepare the message and store it in Bob's mailbox.
- Bob retrieves and read the content of his mailbox at his convenience using a UA.
- Example a person in a (small) company wants to send an email to another person in the same company.
- When the sender and the receiver of an e-mail are on the same system,we need only two user agents.

SECOND SCENARIO:



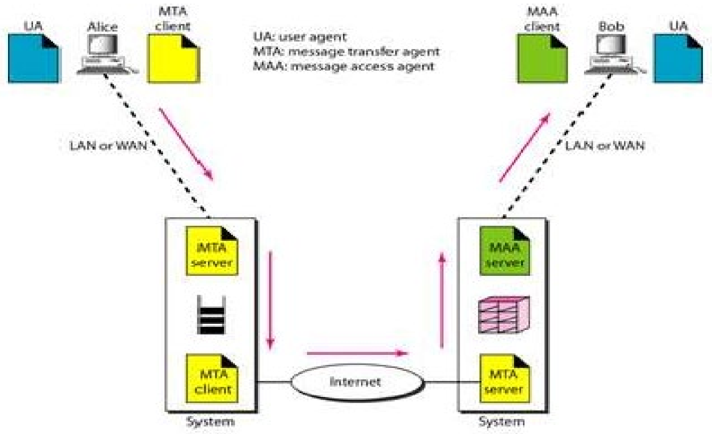
- Alice and Bob are on two different systems but they are directly connected to their systems. The message is sent over the Internet.
- Alice run a UA program to prepare and send the message to the system at her site.
- The system (mail server) at Alice site uses queue to store messages waiting to be sent and runs MTA client to transfer the messages to Bob's system that runs MTA server.
- Bob retrieves and read the content of his mailbox at his convenience using a UA.
- The receiving system must run MTA server process, which needs to run all the time, because it does not know when a client will ask for connection.
- Example: a person at a (small) company wants to send an email to another person in a different (small) company.
- When the sender and the receiver of an e-mail are on different systems,we need two VAs and a pair ofMTAs (client and server).

THIRD SCENARIO:



- Alice is connected to her system via a WAN or LAN.
- Alice run a UA program to prepare the message.
- Alice's message sent through WAN or LAN via a pair of MTA client and server.
- The system (mail server) at Alice site uses queue to store the message and runs MTA client to transfer the messages to Bob's system that runs MTA server.
- Bob retrieves and read the content of his mailbox at his convenience using a UA.
- Example: a person at home wants to send an email to another person in a different organization.
- When the sender is connected to the mail server via a LAN or a WAN,we need two *VAs* and two paIrs of MTAs (clIent and server).

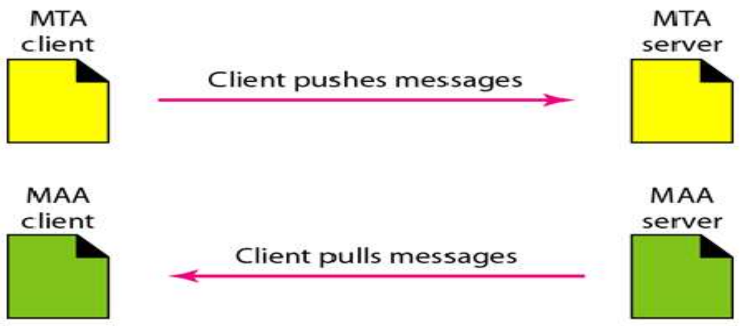
FOURTH SCENARIO:



- Similar to previous scenario, except when the message has arrived at Bob's mail server, Bob uses an MAA to retrieve his message.
- The MAA client sends a request to MAA server.
- Bob cannot bypass the mail server unless he can run MTA server process at all time, which means he needs to connect to the Internet at all time.

- Second, note that Bob needs another pair of client/server programs: message access programs. This is so because an MTA client/server program is a *push* program.The client pushes the message to the server. Bob needs a *pull* program. The client needs to pull the message from the server.

Push versus pull in electronic email



- When both sender and receiver are connected to the mail server via a LAN or a WAN,

we need two VAs, two pairs of MTAs (client and server), and a pair of MAAs
(client and server). This is the most common situation today.

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

PROCESSES:

- Services Provided by a UA
- UA types
- Sending Mail
- Receiving Mail
- Addresses
- Mailing List
- MIME

  - SERVICES PROVIDED BY A UA:

It consists of:

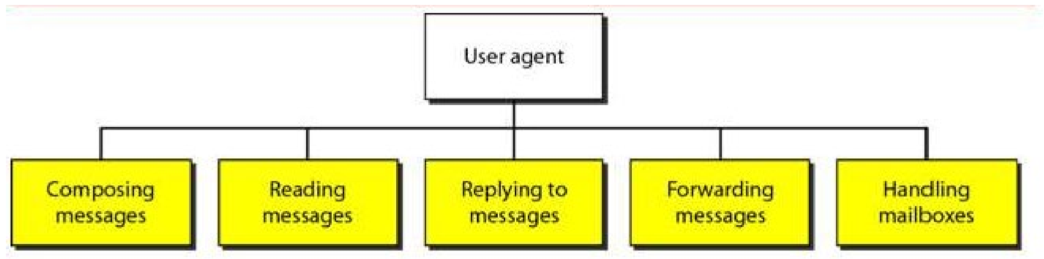
1.composing messages

2.reading messages

3.reply to messages

4.Forwarding messages

5.handling mailboxes



1. COMPOSITE MESSAGES:Composing Messages A user agent helps the user compose the e-mail message to be sent out. Most user agents provide a template on the screen to be filled in by the user. Some even have a built-in editor that can do spell checking, grammar checking, and other tasks expected from a sophisticated word processor. A user, of course, could alternatively use his or her favorite text editor or word processor to create the message and import it, or cut and paste it, into the user agent template.

2. READING MESSAGES:The second duty of the user agent is to read the incoming messages.
When a user invokes a user agent, it first checks the mail in the incoming mailbox.
Most user agents show a one-line summary of each received mail. Each e-mail contains the following fields.
1. A number field.
2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to

3. The size of the message.

4. The sender.

5. The optional subject field.

3. REPLYING TO MESSAGES: After reading a message, a user can use the user agent to reply to a message. A user agent usually allows the user to reply to the original sender or toreply to all recipients of the message. The reply message may contain the original message (for quick reference) and the new message.

4. FORWARDING MESSAGES: *Replying* is defined as sending a message to the sender or recipients of the copy. *Forwarding* is defined as sending the message to a third party. A user agent allows the receiver to forward the message, with or without extra comments, to a third party.

5. HANDLING MAILBOXES: A user agent normally creates two mailboxes: an inbox and an outbox. Each box is a file with a special format that can be handled by the user agent. The inbox keeps all the received e-mails until they are deleted by the user. The outbox keeps all the sent e-mails until the user deletes them. Most user agents today are capable of creating customized mailboxes.

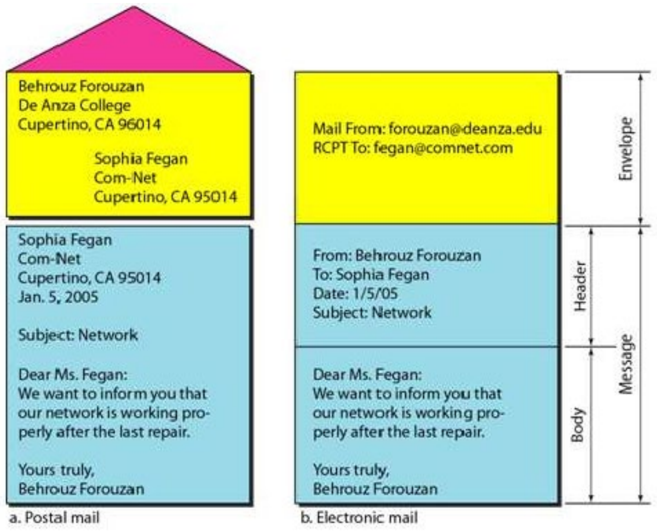
- TYPES OF USER AGENTS:
  There are two types of UA, command-driven and GUI-based.
  COMMAND-DRIVEN:
- A command-driven UA usually accepts a one-character command to perform its task.
- For example, user can type “r” at the command prompt to reply the sender of the message.
- Some examples of command-driven user agents are mail, pine, and elm (used at the introductory period of email).

  GUI-BASED:

- A GUI-based UA has graphical components, such as icons, menu bars, and windows, that allow users to interact with the software by using both keyboard and mouse.
- Some examples of GUI-based user agents are Eudora, Outlook, and Netscape.

- SENDING MAIL:
  User creates mail that looks very similar to postal mail.
  - **Envelope: sender and receiver address**
  - **Message:**

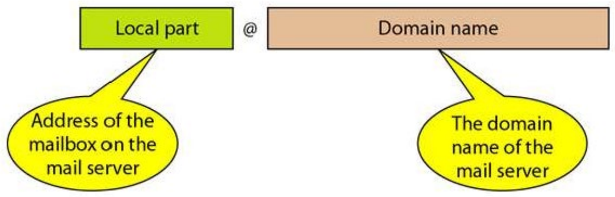
  Header: Define the sender, receiver, subject of the message, ...
  Body: contains the actual information to be read by the recipient.



- RECEIVING MAIL:
- **Email system periodically checks the mailboxes.**
- **If a user has mail, it 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. Summary contains the sender mail address, the subject, and time the mail was sent or received.**

ADDRESS:

■ Mail handling system must use an addressing system with unique addresses.


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

■ Mail handling system must use an addressing system with unique addresses.

■ **Local part:** Name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the user agent.

■ **Domain Name:**

- ■ Organization usually selects one or more hosts to receive and send email; they are sometimes called mail servers or exchangers.
- ■ Domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (e.g., the name of the organization).

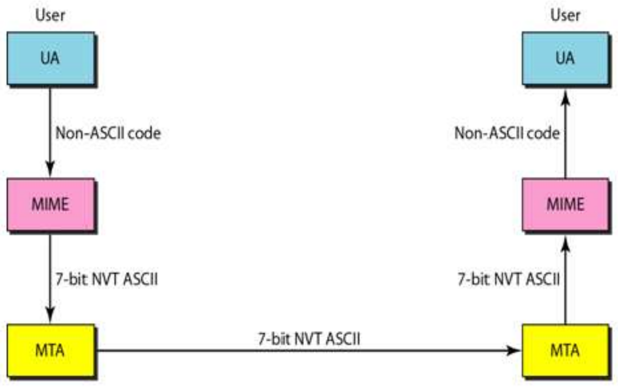
➢ MAILING LIST:

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

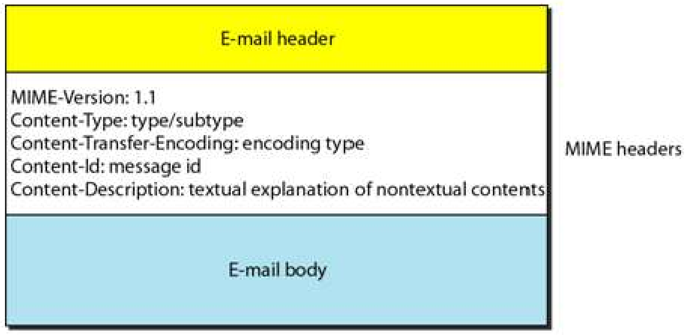
➢ MIME:

Multipurpose Internet Mail Extensions.

- E-mail has a simple structure with limitation.
- The messages can only be sent in NVT-7bit ASCII format.
- We cannot mail in languages supporting more the 7 ASCII bit characters.
- E.g. french, german, japanese etc…
- MIME is a supplementary protocol that allows non-ASCII data to be sent through e-mail.

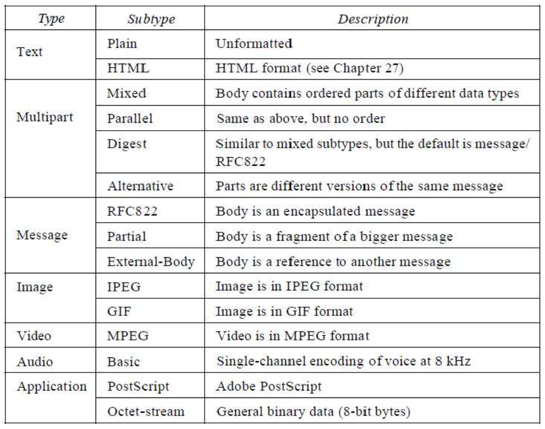


MIME HEADER:


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

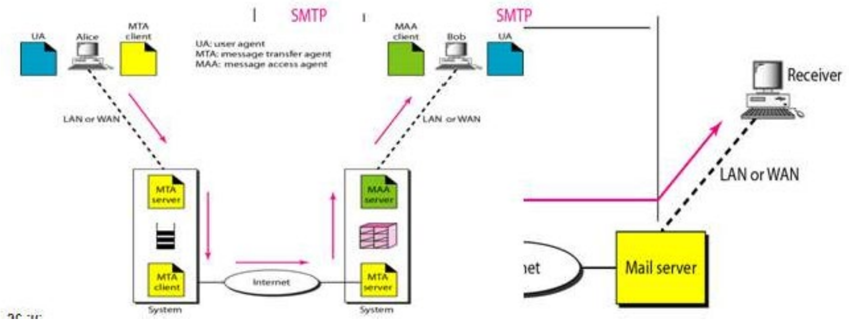
- **MIME-Version:** defines version of the MIME used. Current is 1.1
- **Content-type:** defines the type of data used in the body of the message(7types).
- **Content-transfer-encoding:** defines the method used to encode the message into 0s and 1s for transport (5types).
- **Content-ID:** This header uniquely identifies the whole message in a multiple-message environment.
- **Content-description:** defines whether the body is image, audio, or video.

Types of MIME:



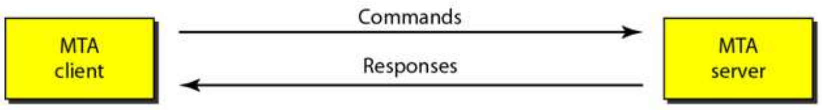
Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Chapter 27)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to mixed subtypes, but the default is message/ RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	IPEG	Image is in IPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

MESSAGE TRANSFER AGENT:SMTP



- The actual mail transfer is done through message transfer agents.
- To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.
- The protocol that is used for MTA client and server in the Internet is SMTP (Simple Mail Transfer Protocol).

- SMTP is used to “push” messages. It is used between the sender and the sender’s mail server and between the two mail servers.
- SMTP only define how commands and responses must be sent back and forth.
- SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.
- Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.



Commands:

- Commands are sent from the client to the server.
- Format: **Keyword: Argument(s)**
- Consists of a keyword followed by zero and more arguments.
- SMTP defines 14 commands.

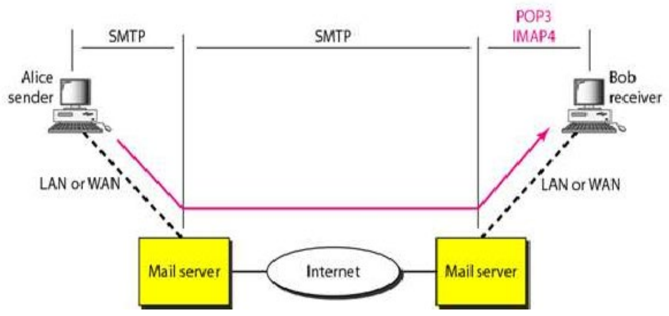
Responses:

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

MAIL TRANSFER PHASES:

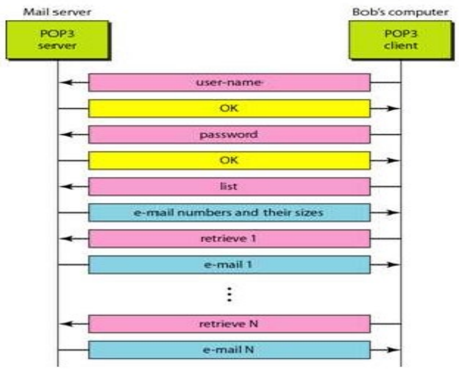
- Process of transferring a mail message occurs in three phases
- Connection establishment
- After a client has made a TCP connection to the well-known port 25, the SMTP server starts the connection phase.
- Message transfer
- After connection establishment, a single message between a sender and one or more recipients can be exchanged
- Connection termination
- After the message is transferred successfully, the client terminates the connection.

➢ MESSAGE ACCESS AGENT:POP AND IMAP



- To retrieve the email, we use a "pull" protocol : the client must pull messages from the server. This stage requires MAA.
- Currently two message access protocols are available:
- POP3 is Post Office Protocol version 3.
- IMAP4 is Internet Mail Access Protocol version 4.

The exchange of commands and responses in POP3:



- Client POP3 s/w is installed on the recipient computer.Server POP3 s/w is installed on the mail server.
- When the user want to download email from the mailbox on the mail server, the POP3 client open connection to POP3 server on TCP port 110.
- The client then send user name and password to access the mailbox. he user asks list and then retrieves the mail messages, one by one.

  ● POP3 has two mode: delete and keep (after retrieval).
- Delete mode: Mail is deleted from the mailbox after each retrieval.
- Keep mode: Mail remains in the mailbox after retrieval**.**

IMAP4:

IMAP4 is similar to POP3 but with some extra features.

Disadvantage of POP3 does not allow user to organize mail on the server; user cannot have different folders on the server; POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides following extra features:

- A user can check the email header prior to downloading.
- A user can search the contents of email before downloading.
- A user can partially download email. This is useful when the bandwidth is limited and when email 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 email storage.

WEB-BASED MAIL:

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Two common sites are Hotmail and Yahoo. The idea is very simple. Mail transfer from Alice's browser to her mail server is done through HTTP.The transfer of the message from the sending mail server to the receiving mail server is still through SMTP. Finally, the message from the receiving server (the Web server) to Bob's browser is done through HTIP. The last phase is very interesting. Instead of POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the e-mail is transferred from the Web server to Bob's browser in HTML format.

FILE TRANFER

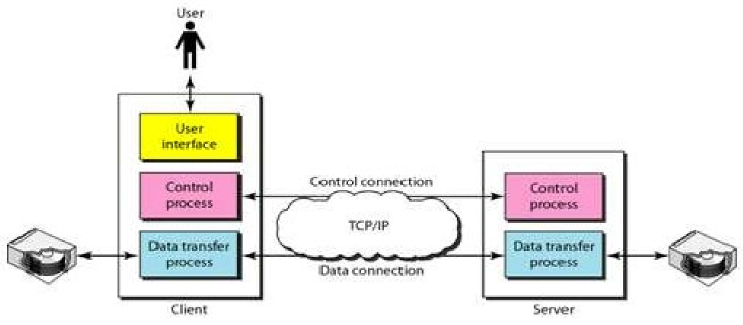
Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to file transfer.

- TRANSFERING OF FILES CONSISTS OF-
  - File transfer protocol(FTP)
    - Communication over control connection
    - Communication over data connection
  - Anonymous FTP

FILE TRANSFR PROTOCOL:is the standard mechanism provided by *TCP/IP for* copying a file from one host to another.

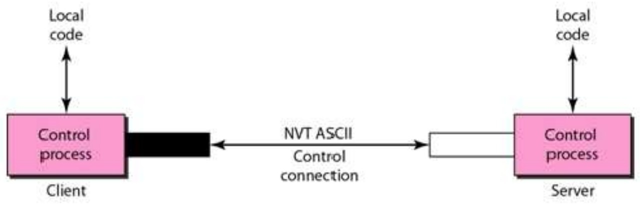
- Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first.Example-two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures.
- All these problems solved by FTP in a very simple and elegant approach.
- FTP differs from other client/server applications in that it establishes two connections between the hosts.
- One connection is used for data transfer, the other for control information (commands and responses).
- Separation of commands and data transfer makes FTP more efficient.
- The control connection uses very simple rules of communication.
- 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 the services of TCP. It needs two TCP connections.

The well-known port 21 is used for the control connection and the well-known port 20 for the data connection.



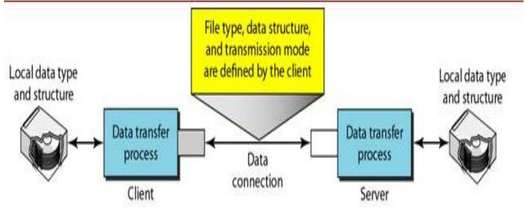
- The basic model of FTP has three components for client and two components for server.
- The control connection remains connected for the entire interactive FTP session.
- The data connection is opened and closed for each file transferred. It opens each time commands that involve transferring files are used and closes when the file is transferred.
- For a single open control connection there will be multiple data connections opened or closed.

  - COMMUNICATION OVER CONTROL CONNECTION:



- FTP uses the same approach as SMTP to communicate across data control connection.
- FTP uses 7-bit ASCII character set.
- Communication is achieved through commands and responses.
- Each command or response is only one short line, so we don't have to worry about file format or file structure.
- Each line is terminated by (carriage return and line feed) end-of-line token.

  - COMMUNICATION OVER DATA CONNECTION:


File type, data structure, and transmission mode are defined by the client
Local data type and structure
Data transfer process
Data connection
Data transfer process
Local data type and structure
Client
Server

- File transfer occurs over the data connection under the control of commands sent over the control connection.
- **RETR** command is used to retrieve a file ☹the file is to be copied from server to client.
- **STOR** command is used to store a file ☹the file is to be copied from client to server.
- **LIST** command is used to list the directories or file names ☹the list is treated as a file and is transferred from server to client.
- The client must define the type of file to be transferred, the structure of data, and the transmission mode. This is done through the control connection before the file is sent, to resolve the heterogeneity problem between two systems.

USING THE DATA CONNECTION:

File type

- **ASCII file**
- **EBCDIC file**
- **Image file**

Data structure

- FTP can transfer a file across the data connection using one of the following interpretations about structure of data:
    - **File structure** (default)
        - File has no structure.
        - Continuous stream of bytes
    - **Record structure**
        - File is divided into records (or structs in C).
        - Used only with text files.
    - **Page structure**
        - File is divided into pages, with each page having a page number and a page header.
        - Pages can be stored or accessed randomly or sequentially.

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

- **Stream mode**
    - Data is delivered from FTP to TCP as a continuous stream of bytes; TCP is responsible for chopping data into segments of appropriate size.
    - End-of-file is closing of data connection by sender.
- **Block mode**
    - Data can be delivered from FTP to TCP in blocks.
    - Block is preceded by 3-byte header.
    - 1st byte is called the block descriptor; next 2 bytes defines the size of block in bytes.
- **Compressed mode**
    - If file is big, data can be compressed.Commonly used compression method is run-length encoding.

➢ **ANONYMOUS FTP:**

- To use FTP, a user needs an account (user name) and a password on the remote server.
- Some sites have a set of files available for public access, to enable anonymous FTP.
- To access these files, a user does not need to have an account or password. Instead, the user can use *anonymous as the user name and guest as the password.*
- User access to the system is very limited. Some sites allow anonymous users only a subset of commands. For example, most sites allow the user to copy some files, but do not allow navigation through the directories.

21. Explain about TCP in detail?

Ans: TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

In TCP we study about the following

- TCP Services
- TCP Features
- Segment
- A TCP Connection
- Flow Control
- Error Control

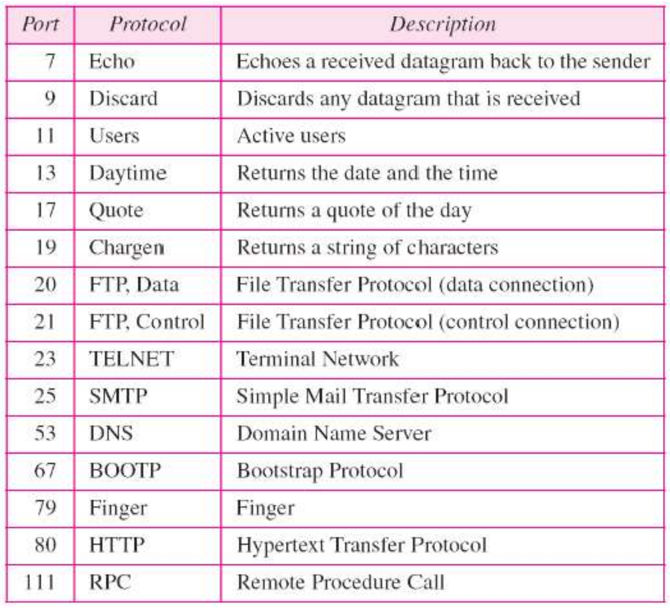
TCP Services

The services offered by TCP to the processes at the application layer are as follows

Process-to-Process Communication

Like UDP, TCP uses port numbers to provide process-to-process communications. Some of the well-known port numbers used by TCP are listed in the below table.

Well-known ports used by TCP

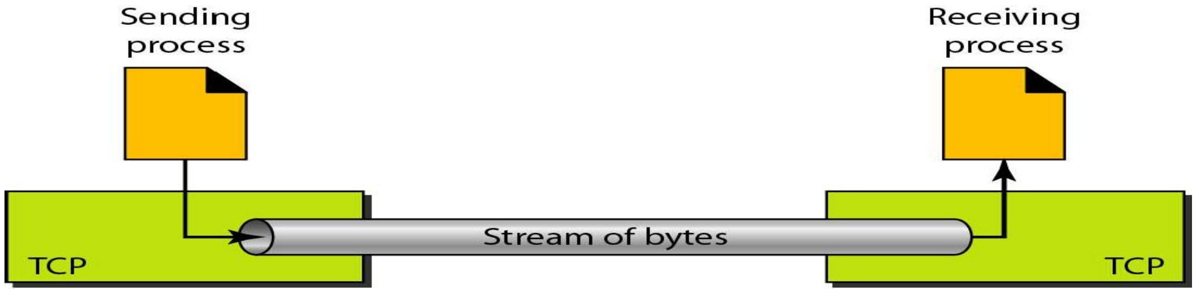


Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
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
111	RPC	Remote Procedure Call

Stream Delivery Service

TCP is a stream-oriented protocol. In TCP, transmission of data occurs as a stream of bytes.TCP creates an environment in which the two processes seem to be connected by an imaginary “tube” through which data transmission takes place. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

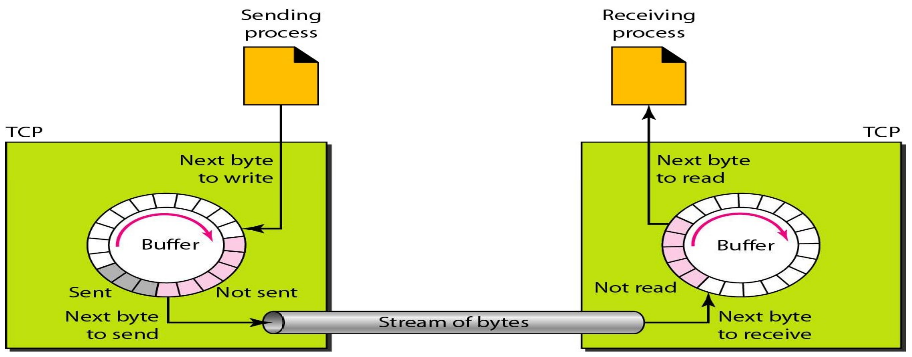
Stream delivery



Sending and Receiving Buffers

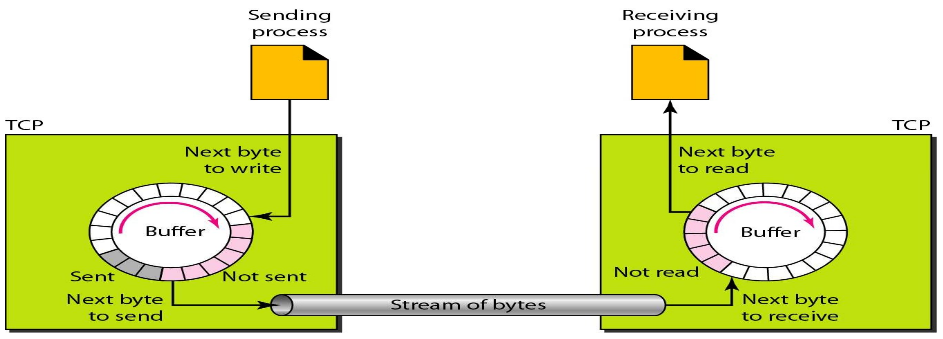
The sender and receiver may not write or read data at the same speed. So, buffers are used for storage. TCP uses two buffers sending buffer and the receiving buffer one for each direction. One way to implement a buffer is to use a circular buffer of 1-byte locations as shown in figure.

Sending and receiving buffers



Segments

IP layer needs to send data in the form of packets. So, TCP groups the stream of bytes into a packet at the transport layer. These packets are called segments.TCP adds a header to each segment and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagram’s and transmitted. This entire operation is transparent to the receiving process.



Full-Duplex Communication

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

Connection Oriented Service

TCP 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.

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.

Reliable Service

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

TCP Features

Features of TCP Protocol:

1. 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.

2. Byte Number

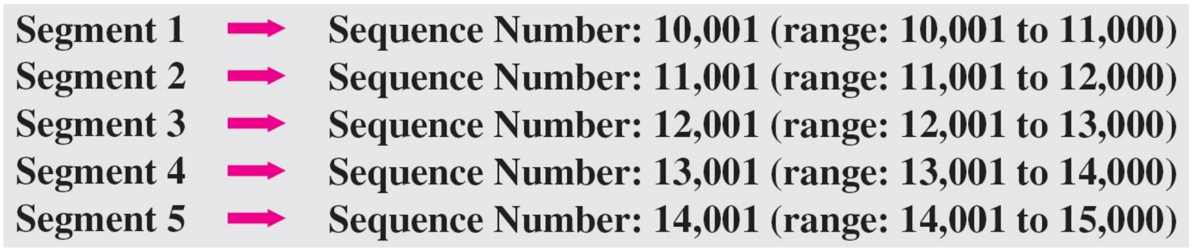
   TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. The numbering does not necessarily start from O. Instead, TCP generates a random number between 0 and 2^{32} 2^{32} - 1 for the number of the first byte.

3. Sequence Number

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

   Example:

   The following shows the sequence number for each segment:



Segment 1	➡	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	➡	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	➡	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	➡	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	➡	Sequence Number: 14,001 (range: 14,001 to 15,000)

   The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

4. Acknowledgement Number

   Communication in TCP is full-duplex; when a connection is established, both parties can send and receive data at the s same time. Each party numbers the bytes, usually with a different starting byte number. The sequence number in each direction shows the number of the first byte carried by the segment. Each party also uses an acknowledgment number to confirm the bytes it has received.

   The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

5. 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.

6. Error Control

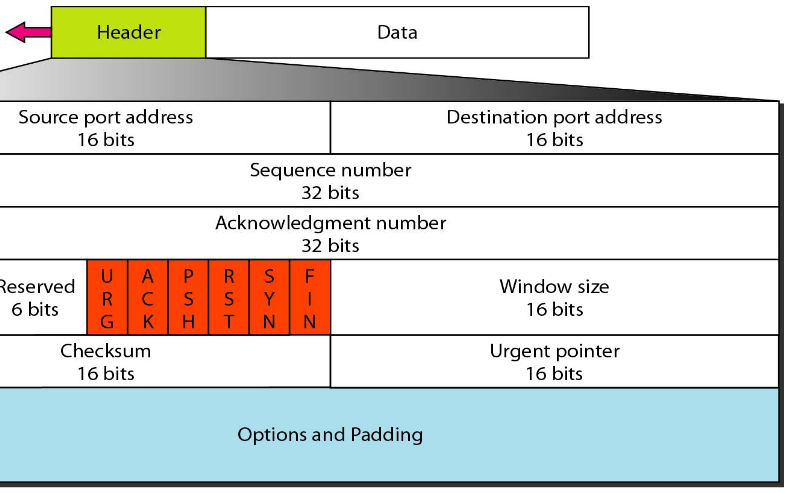
   To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted Segments), error control is byte-oriented.

7. Congestion Control

   TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

Segment

A packet in TCP is called a **Segment**.



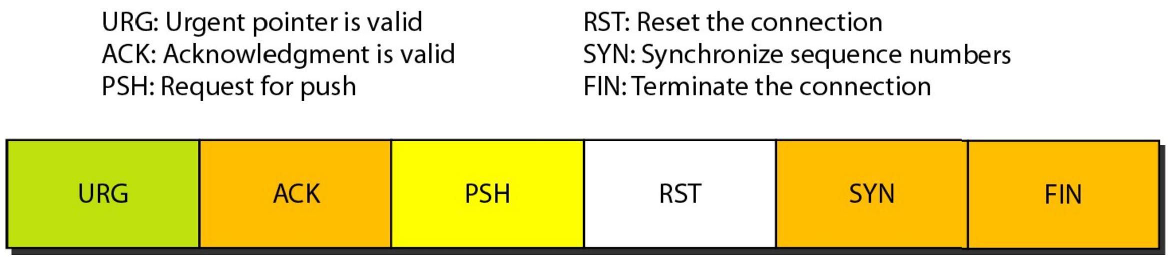
The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

- 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. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.
- Acknowledgment number:
  This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x*+I as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- Header length:

  This 4-bit field indicates the number of 4- byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).

- Reserved
  This is a 6-bit field reserved for future use.
- Control
  This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time.

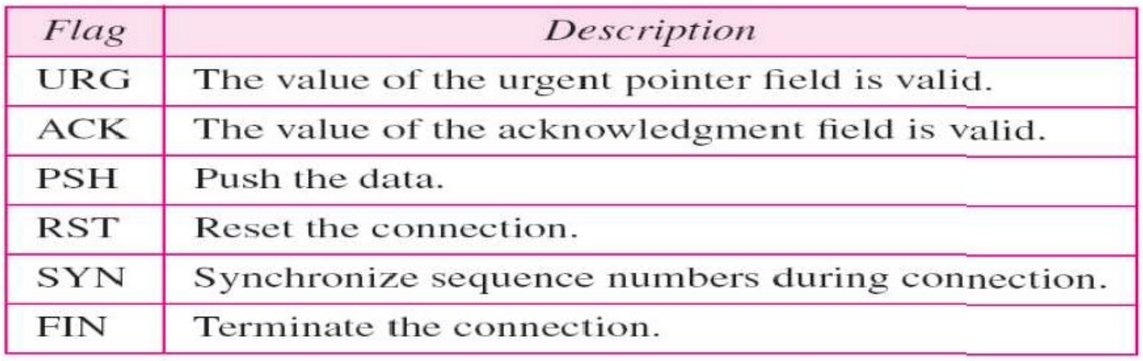
Control field



URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

Description offlags in the control field



Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- Window size:
  This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- Checksum:
  This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.

- Urgent pointer:
  This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- Options:
  There can be up to 40 bytes of optional information in the TCP header.

A TCP Connection

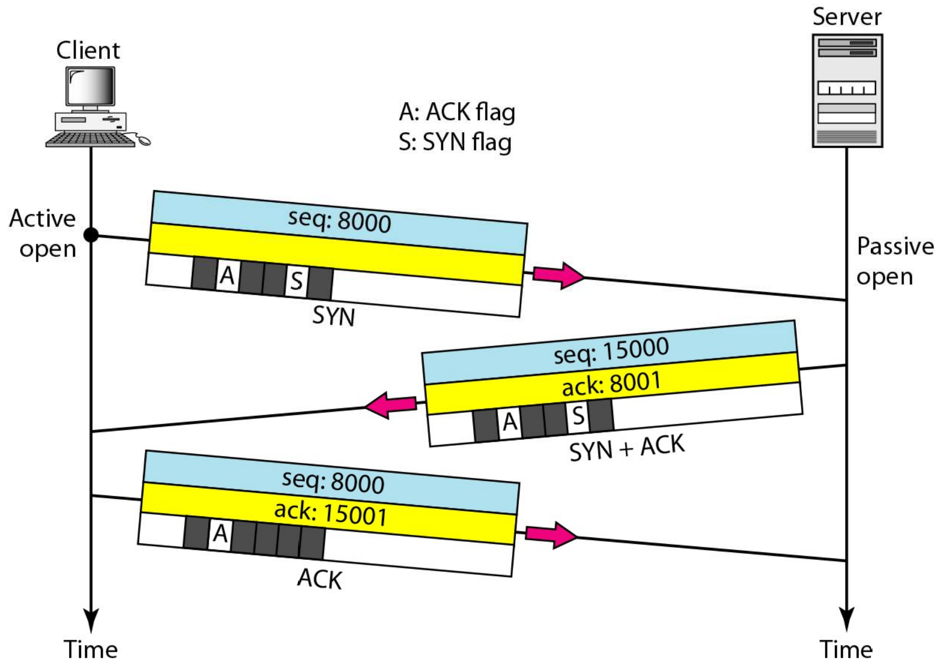
TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. In TCP, connection-oriented transmission requires three phases:

(I) Connection Establishment,
(II) Data Transfer
(III) Connection Termination

Connection Establishment:

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

Connection establishment using three-way handshaking



Three Way Hand Shaking :

The connection establishment in TCP is called **three way handshaking**. In our example, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol.The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a ***passive open***. Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure.

To show the process, we use two time lines: one at each site. Each segment has values for all its header fields and perhaps for some of its option fields, too. However, we show only the few fields necessary to understand each phase. We show the sequence number, the acknowledgment number, the control flags (only those that are set), and the window size, if not empty. The three steps in this phase are as follows.

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.

A SYN segment cannot carry data, but it consumes one sequence number.

1. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

   A SYN +ACK segment cannot carry data, but does consume one sequence number.

2. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one SYN segment; the ACK segment does not consume any sequence numbers.

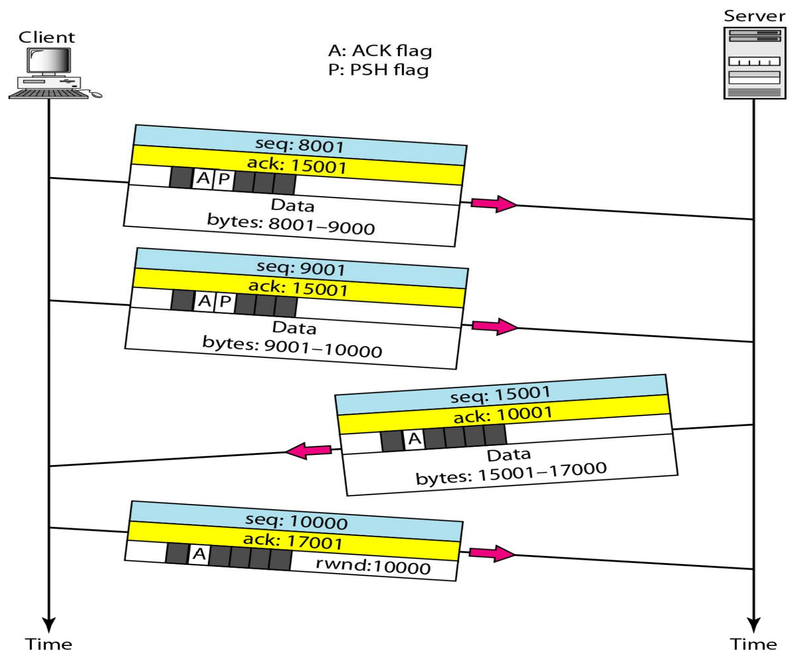
   An ACK segment, if carrying no data, consumes no sequence number.

Simultaneous Open:
A rare situation, called a **simultaneous open**, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

SYN Flooding Attack:
The connection establishment procedure in TCP is susceptible to a serious security problem called the **SYN flooding attack**. This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is corning from a different client by faking the source IP addresses in the datagram's. The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. The TCP server then sends the SYN +ACK segments to the fake clients, which are lost. During this time, however, a lot of resources are occupied without being used. If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash. This SYN flooding attack belongs to a type of security attack known as a **denial-of-service attack**, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

Some implementations of TCP have strategies to alleviate the effects of a SYN attack. Some have imposed a limit on connection requests during a specified period of time. Others filter out datagram's coming from unwanted source addresses. One recent strategy is to postpone resource allocation until the entire connection is set up, using what is called a cookie.

Data Transfer
After connection is established, bidirectional **data transfer** can take place. The client and server can both send data and acknowledgments. The acknowledgment is piggybacked with the data.



In the above example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment.
The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent. Note the values of the sequence and acknowledgment numbers. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received. The segment from the server, on the other hand, does not set the push flag. Most TCP implementations have the option to set or not set this flag.

Pushing Data:

TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size. The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP.
For example, consider an application program that communicates interactively with another application program on the other end. The application program on one site wants to
send a keystroke to the application at the other site and receive an immediate response. Delayed transmission and delayed delivery of data may not be acceptable by the application program.
TCP can handle such a situation. The application program at the sending site can request a *push* operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.
Although the push operation can be requested by the application program, most current implementations ignore such requests. TCP can choose whether or not to use this feature.

Urgent Data:

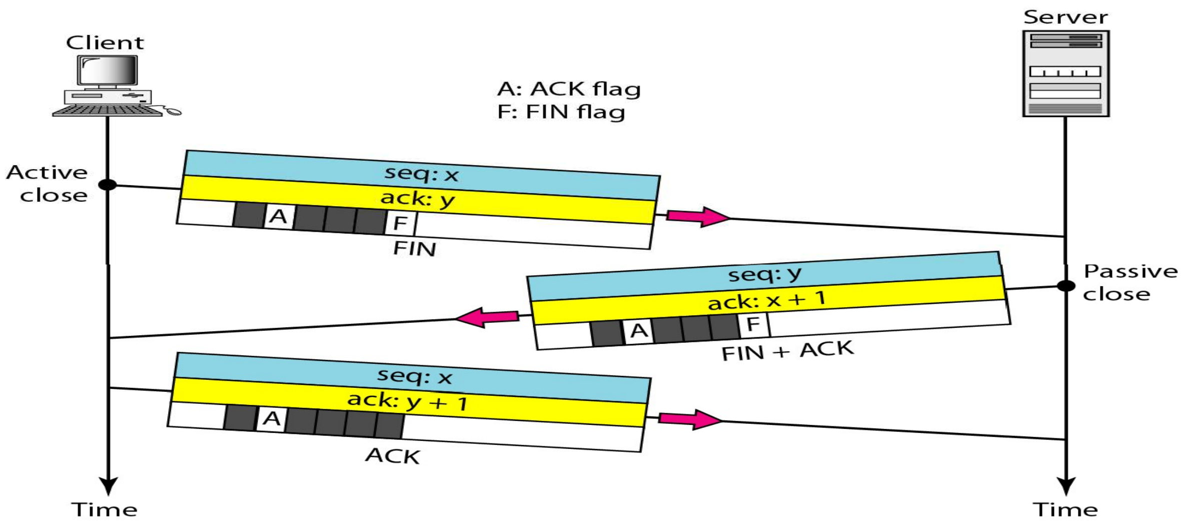
TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, on occasion an application program needs to send *urgent* bytes. This means that the sending application program wants a piece of data to be read out of order by the receiving application program. As an example, suppose that the sending application program is sending data to be processed by the receiving application program. When the result of processing comes back, the sending application program finds that everything is wrong. It wants to abort the process, but it has already sent a huge amount of data. If it issues an abort command (control +C),
these two characters will be stored at the end of the receiving TCP buffer. It will be delivered to the receiving application program after all the data have been processed.

The solution is to send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer. The urgent pointer field in the header defines the end of the urgent data and the start of normal data.
When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

Connection Termination
Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client. Most implementations today allow twooptions for connection termination: three-way handshaking and four-way handshaking
with a half-close option.
Three-Way Handshaking Most implementations today allow *three-way handshaking* for connection termination as shown in Figure

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure. If it is only a control segment, it consumes only one sequence number.
   The FIN segment consumes one sequence number ifit does not carry data

Connection termination using three-way handshaking



.

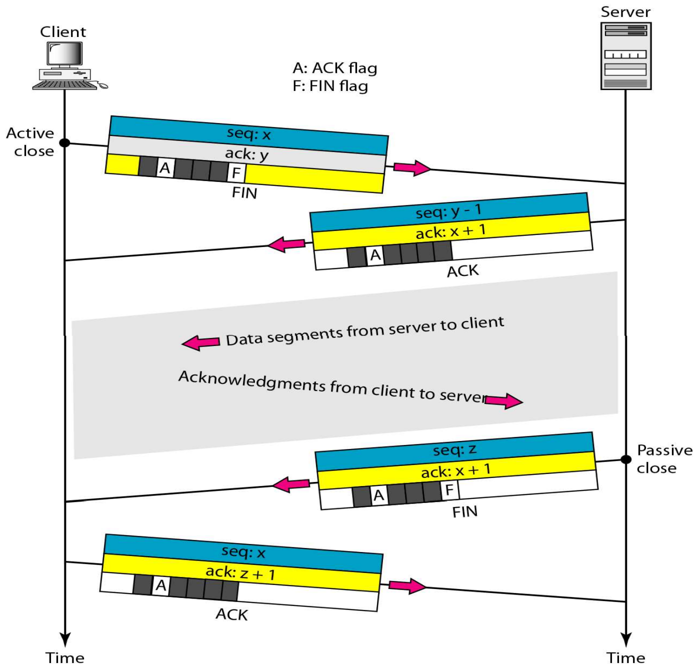
2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN+ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

The FIN +ACK segment consumes one sequence number if it does not carry data.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Half-Close

In TCP, one end can stop sending data while still receiving data. This is called a **half-close**. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin. A good example is sorting. When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all the data, can close the connection in the outbound direction. However, the inbound direction must remain open to receive the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open.



The above figure shows an example of a half-close. The client half-closes the connect on by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops. The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server. Note the sequence numbers we have used. The second segment (ACK) consumes no sequence number. Although the client has received sequence number *y* - 1 and is expecting *y,* the server sequence number is still *y* - 1. When the connection finally closes, the sequence number of the last ACK segment is still *x,* because no sequence numbers are consumed during data transfer in that direction.

Flow Control

TCP uses a sliding window to handle flow control. The sliding window protocol used by TCP is something between the *Go-Back-N* and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive. There are two big differences between this sliding window and the one we used at the data link layer. First, the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented. Second, the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size.

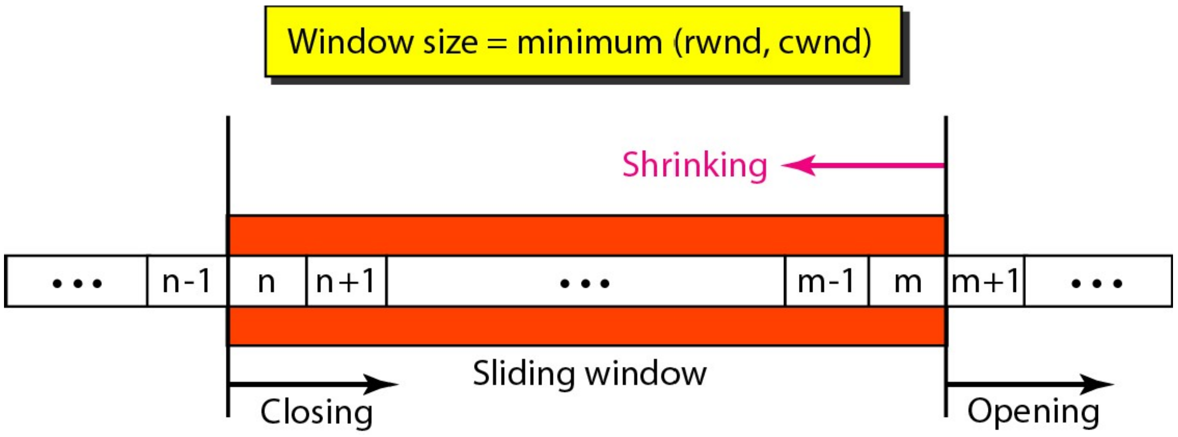


Figure shows the sliding window in TCP. The window spans a portion of the buffer containing bytes received from the process. The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment. The imaginary window has two walls: one left and one right.
The window is *opened, closed,* or *shrunk.* These three activities, are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver in this matter. Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore. Shrinking the window means moving the right wall to the left.

A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented.

The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd)* or *congestion window (cwnd).* The *receiver window* is the value advertisedby the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded. Thecongestion window is a value determined by the network to avoid congestion.
Some points about TCP sliding windows:

- The size of the window is the lesser of *rwnd* and *cwnd.* o The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Error Control

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.
TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.
Checksum
Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment. It cannot be changed for TCP because this would involve reconfiguration of the entire header format.
Acknowledgment
TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

ACK segments do not consume sequence numbers and are not acknowledged.

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

In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.
No retransmission timer is set for an ACK segment.

Retransmission After RTO:
A recent implementation of TCP maintains one **retransmission**
time-out (RTO) timer for all outstanding (sent, but not acknowledged) segments. When the timer matures, the earliest outstanding segment is retransmitted even though lack of a received ACK can be due to a delayed segment, a delayed ACK, or a lost acknowledgment. Note that no time-out timer is set for a segment that carries only an acknowledgment, which means that no such segment is resent. The value of RTO is dynamic in TCP and is updated based on the **round-trip time (RTT)** of segments. An RTI is the time needed for a segment to reach a destination and for an acknowledgment
to be received. It uses a back-off strategy.
Retransmission After Three Duplicate ACK Segments:
The previous rule about retransmission of a segment is sufficient if the value of RTO is not very large. Sometimes,
However, one segment is lost and the receiver receives so many out-of-order segments that they cannot be saved (limited buffer size). To alleviate this situation, most implementations
today follow the three-duplicate-ACKs rule and retransmit the missing segment immediately. This feature is referred to as **fast retransmission**
Out-of-Order Segments
When a segment is delayed, lost, or discarded, the segments following that segment arrive out of order. Originally, TCP was designed to discard all out-of-order segments, resulting in the retransmission of the missing segment and the following segments. Most implementations today do not discard the out-of-order

segments. They store them temporarily and flag them as out-of-order segments until the missing segment arrives. However, that the out-of-order segments are not delivered to the process. TCP guarantees that data are delivered to the process in order

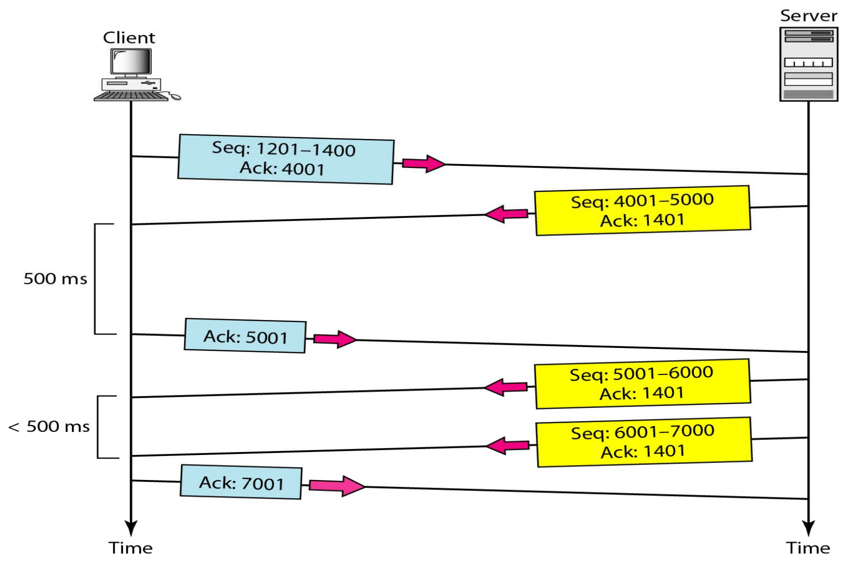
Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.

Some Scenarios

some examples of scenarios that occur during the operation of
TCP.

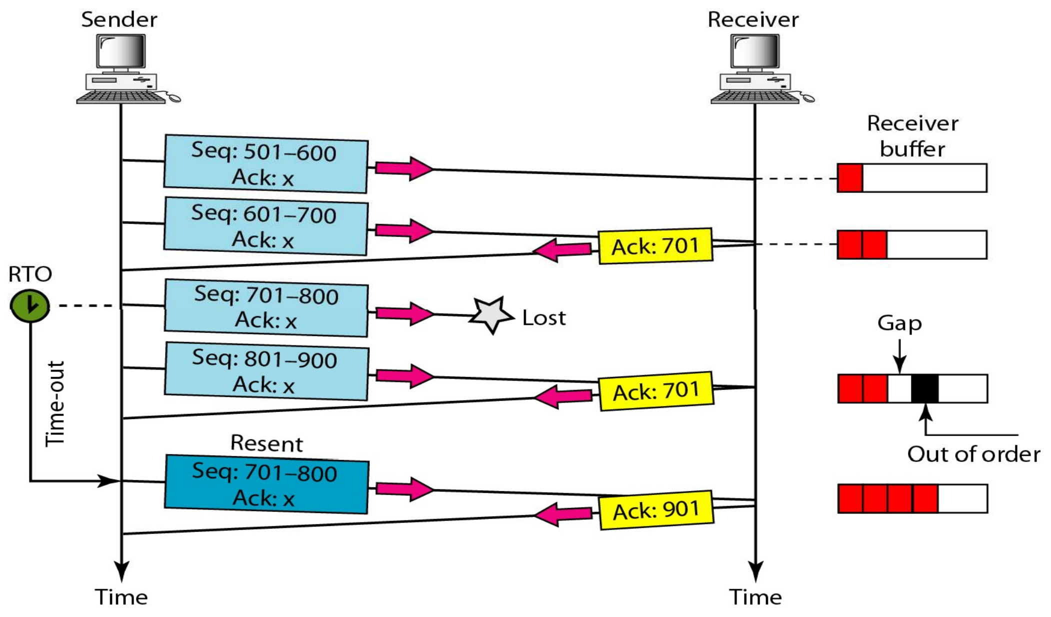
Normal Operation

The first scenario shows bidirectional data transfer between two systems, as shown in Figure. The client TCP sends one segment; the server TCP sends three. The figure shows which rule applies to each acknowledgment. There are data to
be sent, so the segment displays the next byte expected. When the client receives the first segment from the server, it does not have any more data to send; it sends only an ACK segment. However, the acknowledgment needs to be delayed for 500 ms to see if any more segments arrive. When the timer matures, it triggers an acknowledgment. This is so because the client has no knowledge if other segments are coming; it cannot delay the acknowledgment forever. When the next segment arrives, another acknowledgment timer is set. However, before it matures, the third segment arrives. The arrival of the third segment triggers another acknowledgment.



Lost Segment

In this scenario, we show what happens when a segment is lost or corrupted. A lost segment and a corrupted segment are treated the same way by the receiver. A lost segment is discarded somewhere in the network; a corrupted segment is discarded by the receiver itself. Both are considered lost. Figure shows a situation in which a segment is lost and discarded by some router in the network, perhaps due to congestion.

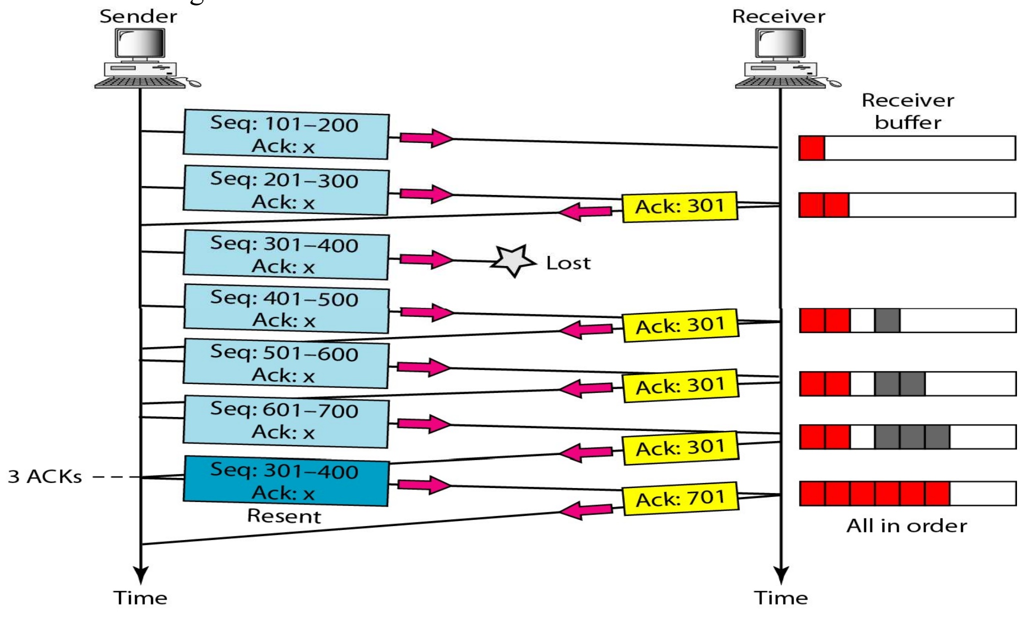


We are assuming that data transfer is unidirectional: one site is sending, the other is receiving. In our scenario, the sender sends segments 1 and 2, which are acknowledged immediately by an ACK. Segment 3, however, is lost. The receiver receives segment 4, which is out of order. The receiver stores the data in the segment in its buffer but leaves a gap to indicate that there is no continuity in the data. The receiver immediately sends an acknowledgment to the sender, displaying the next byte it expects. The receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.

The receiver TCP delivers only ordered data to the process.

Fast Retransmission

Inthis scenario, we want to show the idea of fast retransmission.Our scenario is the same as the second except that the RTO has a higher value

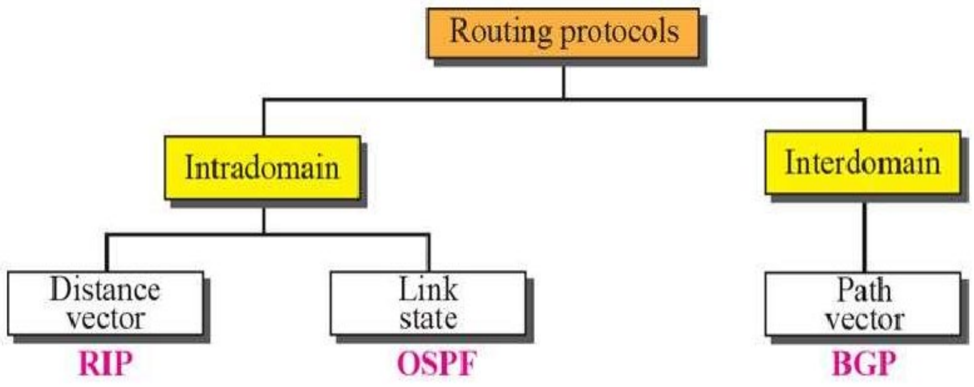


When the receiver receives the fourth, fifth, and sixth segments, it triggers an acknowledgment. The sender receives four acknowledgments with the same value (three duplicates). Although the timer for segment 3 has not matured yet, the fast transmission requires that segment 3, the segment that is expected by all these acknowledgments, be resent immediately. Only one segment is retransmitted although four segments are not acknowledged. When the sender receives the retransmitted ACK, it knows that the four segments are safe and sound because acknowledgment is cumulative.

22. WITH NEAT DIAGRAMS DISCUSS ABOUT THE 3 UNICAST ROUTING PROTOCOLS?

ANS. **UNICAST ROUTING PROTOCOLS**

- A **routing protocol** is a combination of rules and procedures that let routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhood.
- The routing protocols also include procedures for combining information received from other routers.

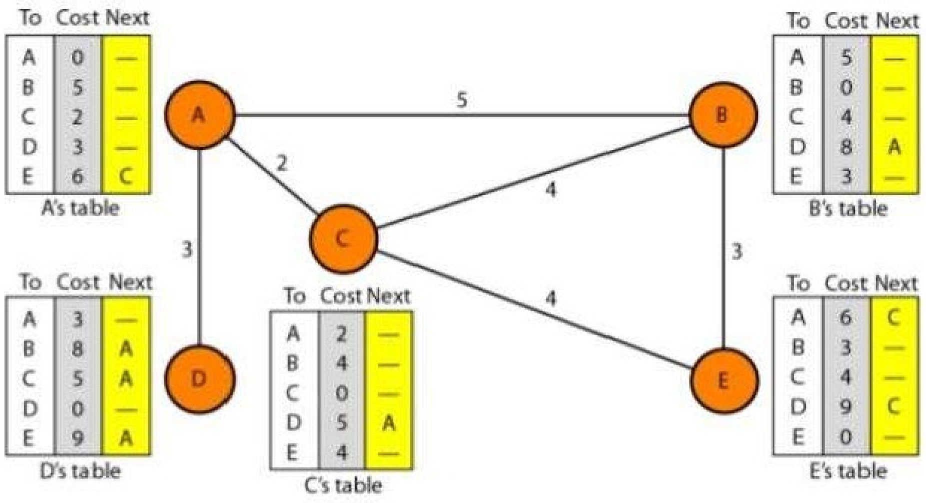


Intra- and Interdomain Routing

- Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems.
- An **autonomous system**(AS) is a group of networks and routers under the authority of a single administration.
- 1. Routing inside an autonomous system is referred to as **intradomain routing**.
  2. Routing between autonomous systems is referred to as **interdomain routing**.

Distance Vector Routing

- Each node maintains a vector (table) of minimum distances to every node.
- The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing).
- We can think of nodes as the cities in an area and the lines as the roads connecting them. A table can show a tourist the minimum distance between cities.



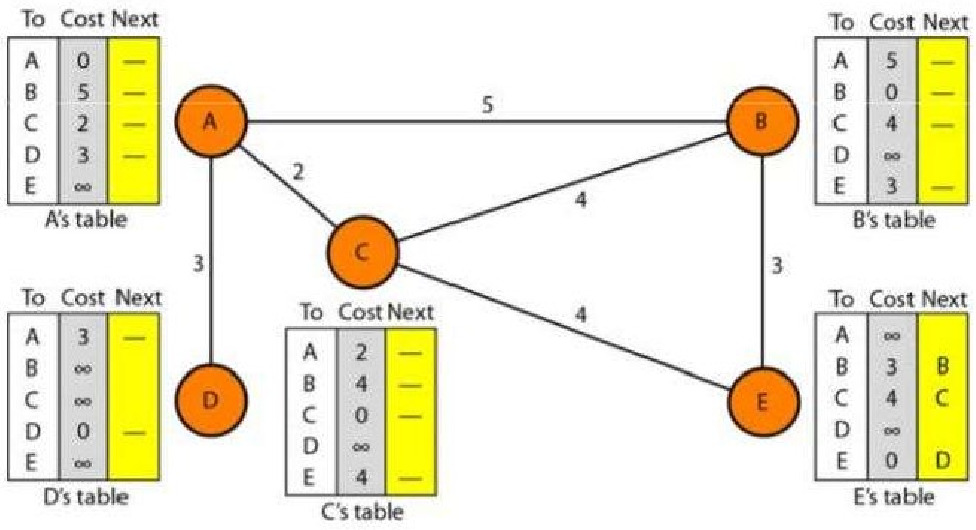
A's table: To Cost Next — A 0 —; B 5 —; C 2 —; D 3 —; E 6 C
B's table: To Cost Next — A 5 —; B 0 —; C 4 —; D 8 A; E 3 —
C's table: To Cost Next — A 2 —; B 4 —; C 0 —; D 5 A; E 4 —
D's table: To Cost Next — A 3 —; B 8 A; C 5 A; D 0 —; E 9 A
E's table: To Cost Next — A 6 C; B 3 —; C 4 —; D 9 C; E 0 —
A B C D E
5 2 3 4 3 4

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

Initialization

- The tables in Figure 22.14 are stable; each node knows how to reach any other node and the cost.
- At the beginning, however, this is not the case. Each node can know only the distance between itself and its **immediate neighbors**, those directly connected to it.
- So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors.
- The distance for any entry that is not a neighbor is marked as infinite (unreachable).

Initialization of tables in DV Routing



A's table: To Cost Next — A 0 —; B 5 —; C 2 —; D 3 —; E ∞
B's table: To Cost Next — A 5 —; B 0 —; C 4 —; D ∞; E 3 —
C's table: To Cost Next — A 2 —; B 4 —; C 0 —; D ∞; E 4 —
D's table: To Cost Next — A 3 —; B ∞; C ∞; D 0 —; E ∞
E's table: To Cost Next — A ∞; B 3 B; C 4 C; D ∞; E 0 D
A B C D E
5 2 3 4 3 4

Sharing

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

- There is only one problem. How much of the table must be shared with each neighbor?
- The best solution for each node is to send its entire table to the neighbor and let the neighbor decide what part to use and what part to discard.
- Sharing here means sharing only the first two columns.
- If any of the rows can be used, the next node is the sender of the table.

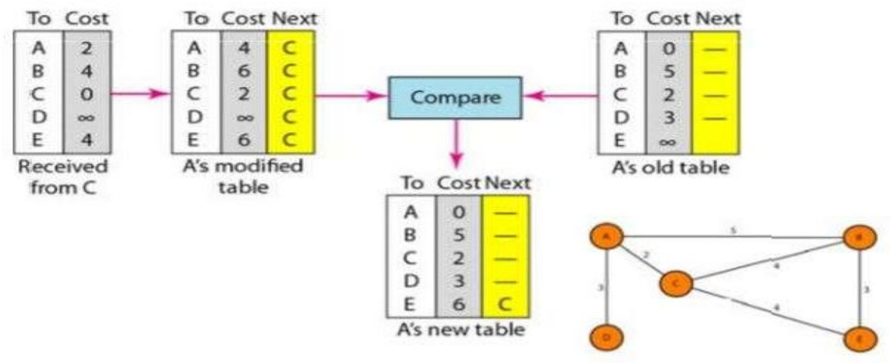
In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

Updating

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

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column.
2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
   a. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is a tie, the old one is kept.
   b. If the next-node entry is the same, the receiving node chooses the new row.

Updating in Distance Vector Routing



To	Cost
A	2
B	4
C	0
D	∞
E	4

Received from C

To	Cost	Next
A	4	C
B	6	C
C	2	C
D	∞	C
E	6	C

A's modified table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

A's old table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

A's new table

When to Share

- The table is sent both periodically and when there is a change in the table.
- **1. Periodic Update:** A node sends its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

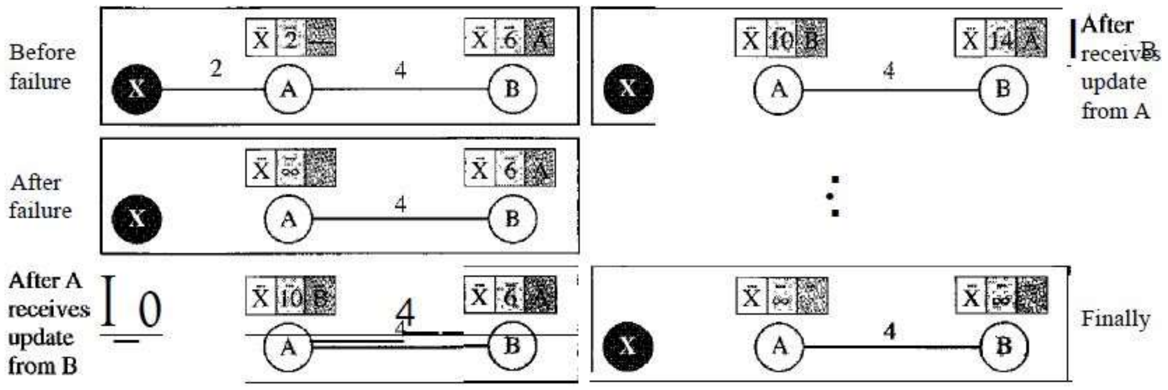
2. Triggered Update: A node sends its two-column routing table to its neighbors anytime there is a change in its routing table. This is called a triggered update. The change can result from the following.

a. A node receives a table from a neighbor, resulting in changes in its own table after updating.

b. A node detects some failure in the neighboring links which results in a distance change to infinity.

Two-Node Loop Instability

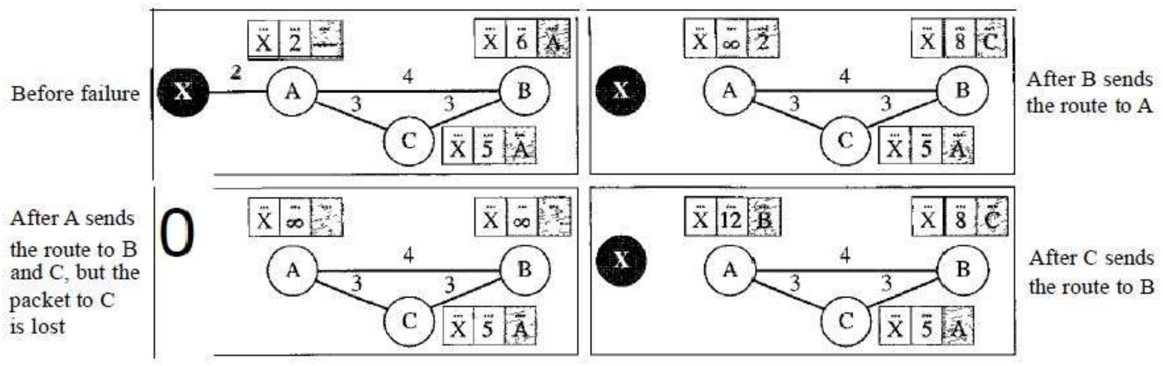
- A problem with distance vector routing is network instability.


Before failure
After failure
After A receives update from B
After B receives update from A
Finally
X
A
B
2
4

- At the beginning, both nodes A and B know how to reach node X.
- But suddenly, the link between A and X fails. Node A changes its table. If A can send its table to B immediately, everything is fine.
- However, the system becomes unstable if B sends its routing table to A before receiving A's routing table.
- Based on the triggered update strategy, A sends its new update to B. Now B thinks that something has been changed around A and updates its routing table.
- The cost of reaching X increases gradually until it reaches infinity.
- Node A thinks that the route to X is via B; node B thinks that the route to X is via A.
- If A receives a packet destined for X, it goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem.
- A few solutions have been proposed for instability of this kind.

1. Defining Infinity:
- The first obvious solution is to redefine infinity to a smaller number, such as 100.
- As a matter of fact, most implementations of the distance vector protocol define the distance between each node to be I and define 16 as infinity.
2. Split Horizon:
- Instead of flooding the table through each interface.
- If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has corne from A (A already knows).
- Node A keeps the value of infinity as the distance to X. Later when node A sends its routing table to B, node B also corrects its routing table.
- The system becomes stable after the first update: both node A and B know that X is not reachable.
3. Split Horizon and Poison Reverse:
- Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table.
- When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently.
- The split horizon strategy can be combined with the **poison reverse** strategy.
- Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

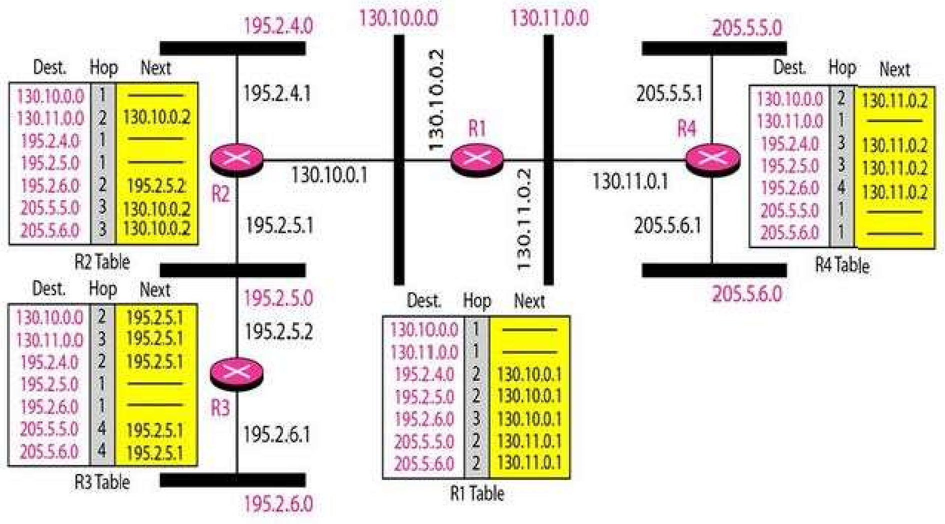
Three-Node Instability
- If the instability is between three nodes, stability cannot be guaranteed.


Before failure
After B sends the route to A
After A sends the route to B and C, but the packet to C is lost
After C sends the route to B

- After finding that X is not reachable, node A sends a packet to Band C.
- Node B immediately updates its table, but the packet to C is lost in the network and never reaches C.
- Node C still thinks that there is a route to X via A with a distance of 5.
- After a while, node C sends to Bits routing table, which includes the route to X. Node B is totally fooled here.
- Node B updates its table, showing the route to X via C with a cost of 8.
- So after awhile node B may advertise this route to A.
- Now A is fooled and updates its table to show that A can reach X via B with a cost of 12. The loop continues leading to three-node instability.

RIP

- The Routing Information Protocol (RIP) is an intradomain routing protocol used inside an autonomous system based on distance vector routing.
- RIP implements distance vector routing directly with some considerations:
  1. In an autonomous system, we are dealing with routers and networks (links). The routers have routing tables; networks do not.
  2. The destination in a routing table is a network, which means the first column defines a network address.
  3. The metric in RIP is called a hop count. The distance is defined as the number of links (networks) to reach the destination.
  4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
  5. The next-node column defines the address of the router to which the packet is to be sent to reach its destination.



R2 Table

Dest.	Hop	Next
130.10.0.0	1	———
130.11.0.0	2	130.10.0.2
195.2.4.0	1	———
195.2.5.0	1	———
195.2.6.0	2	195.2.5.2
205.5.5.0	3	130.10.0.2
205.5.6.0	3	130.10.0.2

R4 Table

Dest.	Hop	Next
130.10.0.0	2	130.11.0.2
130.11.0.0	1	———
195.2.4.0	3	130.11.0.2
195.2.5.0	3	130.11.0.2
195.2.6.0	4	130.11.0.2
205.5.5.0	1	———
205.5.6.0	1	———

R3 Table

Dest.	Hop	Next
130.10.0.0	2	195.2.5.1
130.11.0.0	3	195.2.5.1
195.2.4.0	2	195.2.5.1
195.2.5.0	1	———
195.2.6.0	1	———
205.5.5.0	4	195.2.5.1
205.5.6.0	4	195.2.5.1

R1 Table

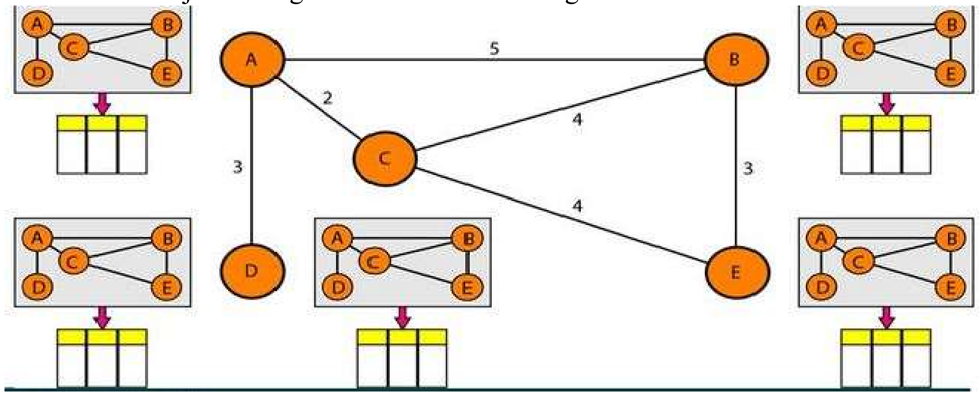
Dest.	Hop	Next
130.10.0.0	1	———
130.11.0.0	1	———
195.2.4.0	2	130.10.0.1
195.2.5.0	2	130.10.0.1
195.2.6.0	3	130.10.0.1
205.5.5.0	2	130.11.0.1
205.5.6.0	2	130.11.0.1

- To send a packet to one of the three networks at the far left, router Rl needs to deliver the packet to R2. The next-node entry for these three networks is the interface of router R2 with IP address 130.10.0.1.

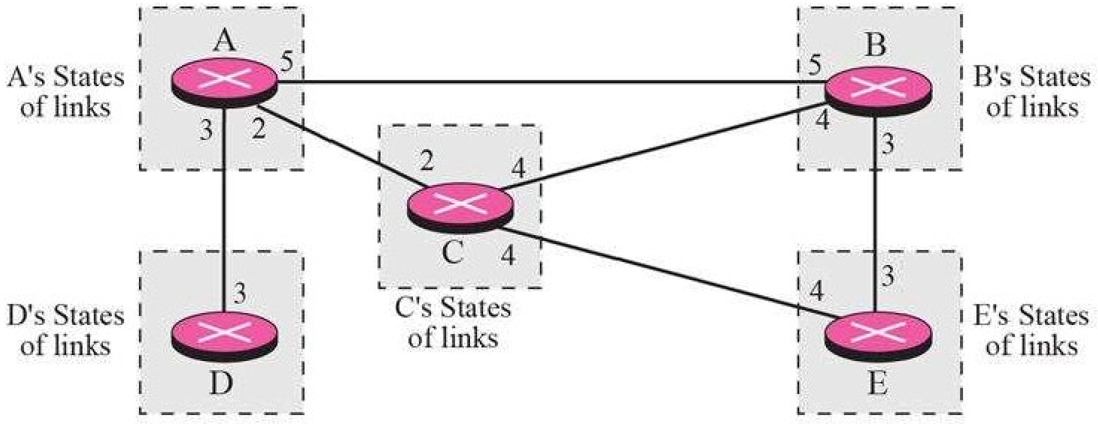
- To send a packet to the two networks at the far right, router Rl needs to send the packet to the interface of router R4 with IP address 130.11.0.1.

Link State Routing

- In link state routing, if each node in the domain has the entire topology of the domain the list of nodes and links, how they are connected including the type, cost (metric), and condition of the links (up or down)-the node can use Dijkstra's algorithm to build a routing table.



- Each node in the domain has the entire topology of the domain.
- Each node uses the same topology to create the routing tables.
- The topology must be dynamic, representing the latest state of each node and each link.
- How can a common topology be dynamic and stored in each node? No node can know the topology at the beginning or after a change somewhere in the network.
- Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links.



- Node A knows that it is connected to node B with metric 5, to node C with metric 2, and so on.

Building Routing Tables

In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.
1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding,** in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP)

- A link state packet can carry a some amount of information: the node identity, the list of links, a sequence number, and age.
- The first two, node identity and the list of links, are needed to make the topology.
- The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones.
- The fourth, age, prevents old LSPs from remaining in the domain for a long time.
- LSPs are generated on two occasions:
  1. *When there is a change in the topology of the domain.*
  2. *On a periodic basis:* A longer period ensures that flooding does not create too much traffic on the network.

Flooding of LSPs

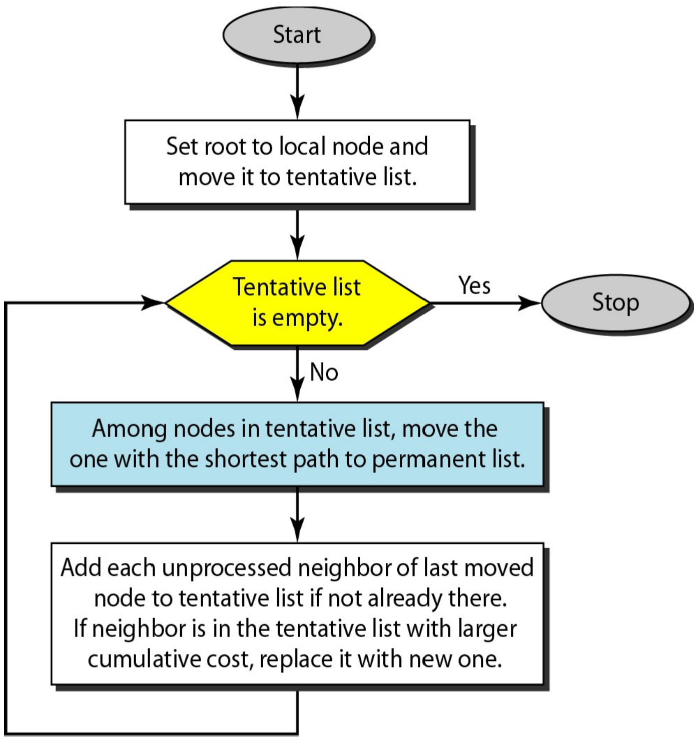
- After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors called flooding which is based on the following:
  1. The creating node sends a copy of the LSP out of each interface.
  2. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:

  a. It discards the old LSP and keeps the new one.

  b. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain.

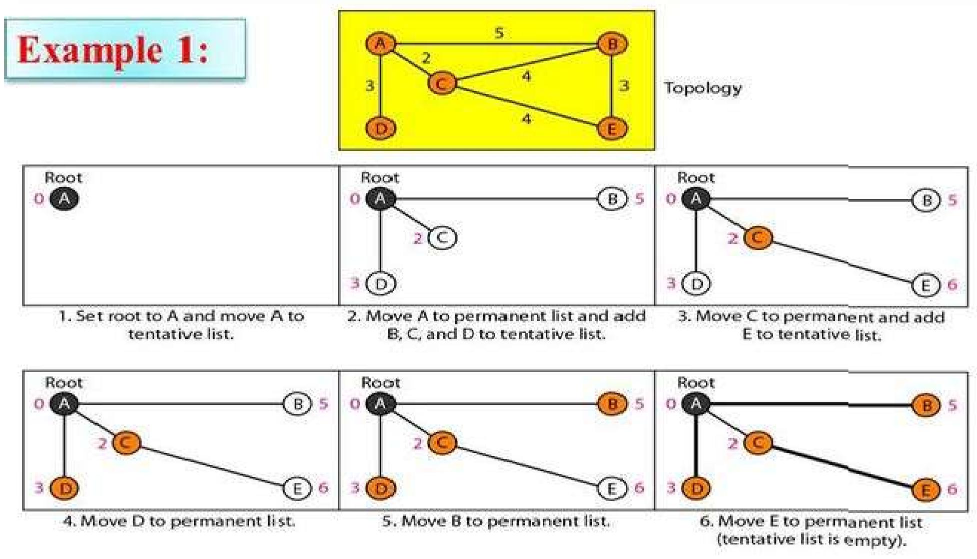
Formation of Shortest Path Tree: Dijkstra Algorithm

- A shortest path tree is a tree in which the path between the root and every other node is the shortest.
- The Dijkstra algorithm creates a shortest path tree from a graph.
- The algorithm divides the nodes into two sets: tentative and permanent.
- It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.


Start
Set root to local node and move it to tentative list.
Tentative list is empty.
Yes
Stop
No
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there.
If neighbor is in the tentative list with larger cumulative cost, replace it with new one.

- Example of formation of shortest path tree:

Example 1:


Topology
Root
1. Set root to A and move A to tentative list.
2. Move A to permanent list and add B, C, and D to tentative list.
3. Move C to permanent and add E to tentative list.
4. Move D to permanent list.
5. Move B to permanent list.
6. Move E to permanent list (tentative list is empty).

1. Permanent list: empty Tentative list: A(O)
2. Permanent list: A(O) Tentative list: B(5), C(2), D(3)
3. Permanent list: A(O), C(2) Tentative list: B(5), 0(3), E(6)
4. Permanent list: A(O), C(2), D(3) Tentative list: B(5), E(6)
5. Permanent list: A(O), B(5), C(2), D(3) Tentative list: E(6)
6. Permanent list: A(O), B(5), C(2), D(3), E(6) Tentative list: empty

<u>Calculation of Routing Table from Shortest Path Tree</u>

- Each node uses the shortest path tree protocol to construct its routing table which shows the cost of reaching each node from the root.

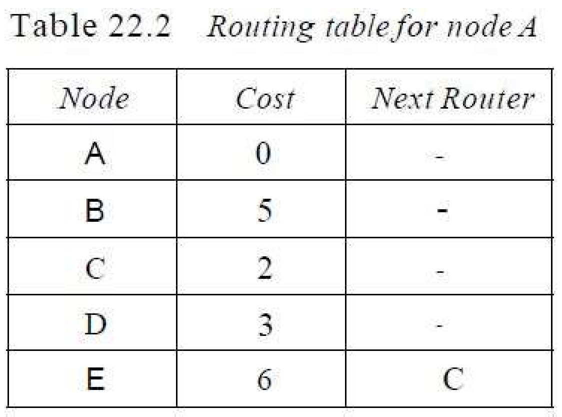


Table 22.2 *Routing table for node A*

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

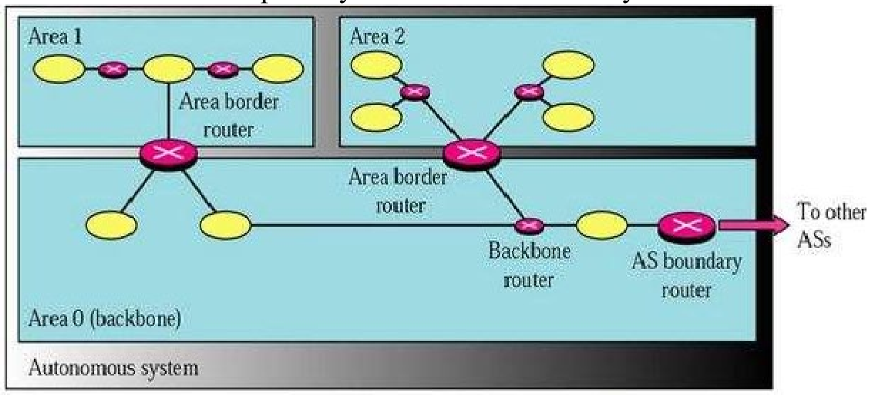
OSPF

- The Open Shortest Path First or **OSPF protocol** is an intradomain routing protocol based on link state routing.

Areas

- An **area** is a collection of networks, hosts, and routers all contained within an autonomous system.
- Routers inside an area flood the area with routing information. At the border of an area, special routers called **area border routers** summarize the information about the area and send it to other areas.
- All the areas inside an autonomous system must be connected to the backbone.

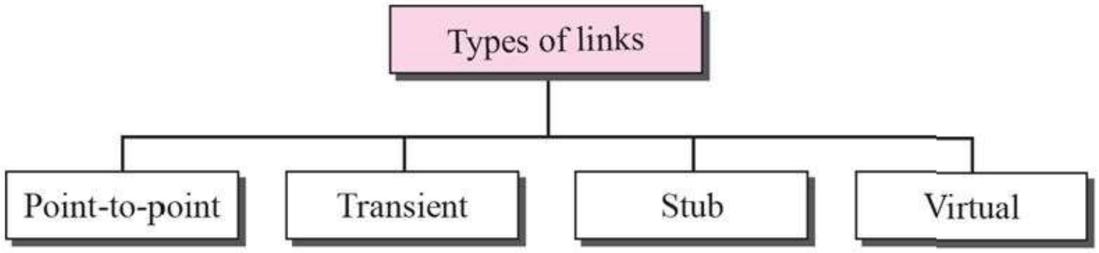
- The routers inside the backbone are called the **backbone routers**.
- A virtual link between routers must be created by an administrator to allow continuity of the functions of the backbone as the primary area if the connectivity fails.



Metric: Type of Service

Types of Links:

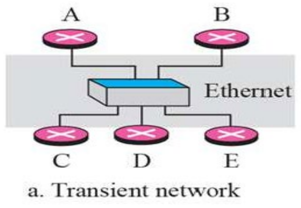
- In OSPF terminology, a connection is called a *link.*



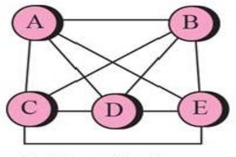
1. A point-to-point link connects two routers without any other host or router in between. Example: two routers connected by a telephone line or a T line.

a. Point-to-point network

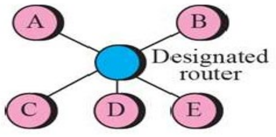
2. A transient link is a network with several routers attached to it. The data can enter through any of the routers and leave through any router. consider the Ethernet in Figure 22.27a. Router A has routers B, C, D, and E as neighbors. Router B has routers A, C, D, and E as neighbors. If we want to show the neighborhood relationship in this situation, we have the graph shown in Figure 22.27b.



a. Transient network

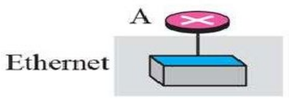


b. Unrealistic

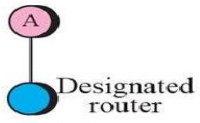


c. Realistic

3. Stub link: The data packets enter the network through a single router and leave the network through the same router. The link is only one-directional.

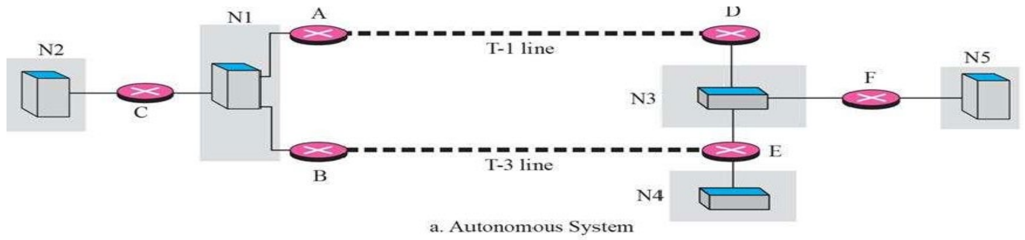


a. Stub network

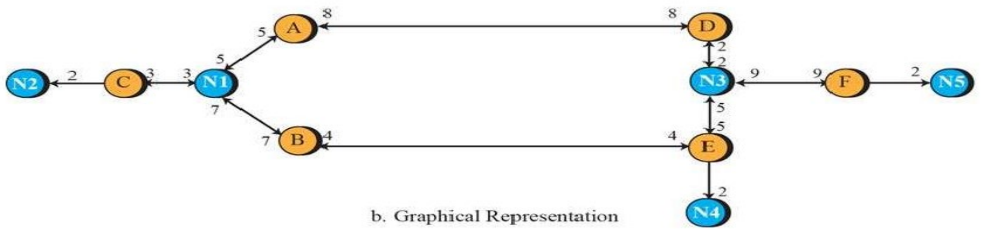


b. Representation

Graphical Representation



a. Autonomous System



b. Graphical Representation

Path Vector Routing

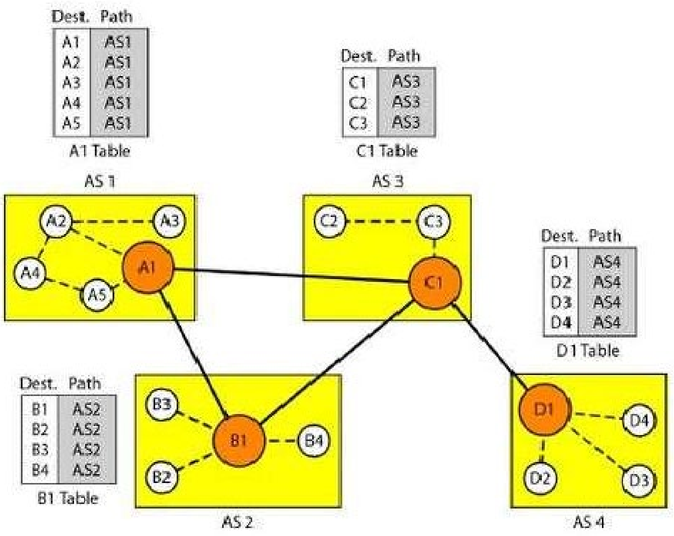
- Path vector routing is an inter-domain routing protocol.
- We assume that there is one node (there can be more, but one is enough for our conceptual

  discussion) in each autonomous system that acts on behalf of the entire autonomous system. Let us call it the speaker node.
- The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs.

Initialization

- At the beginning, each speaker node can know only the reachability of nodes inside its autonomous system.

Figure 22.30 ***Initial routing tables in path vector routing***



Sharing

- A speaker in an autonomous system shares its table with immediate neighbors.

Updating

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

Dest.	Path
A1	AS1
...	
A5	AS1
B1	AS1-AS2
...	...
B4	AS1-AS2
C1	AS1-AS3
...	...
C3	AS1-AS3
D1	AS1-AS2-AS4
...	...
D4	AS1-AS2-AS4

A1 Table

Dest.	Path
A1	AS2-AS1
...	
A5	AS2-AS1
B1	AS2
...	...
B4	AS2
C1	AS2-AS3
...	...
C3	AS2-AS3
D1	AS2-AS3-AS4
...	...
D4	AS2-AS3-AS4

B1 Table

Dest.	Path
A1	AS3-AS1
...	
A5	AS3-AS1
B1	AS3-AS2
...	...
B4	AS3-AS2
C1	AS3
...	...
C3	AS3
D1	AS3-AS4
...	...
D4	AS3-AS4

C1 Table

Dest.	Path
A1	AS4-AS3-AS1
...	
A5	AS4-AS3-AS1
B1	AS4-AS3-AS2
...	...
B4	AS4-AS3-AS2
C1	AS4-AS3
...	...
C3	AS4-AS3
D1	AS4
...	...
D4	AS4

D1 Table

- ▪ Loop prevention:
  When a router receives a message, it checks to see if its autonomous system is in the path list to the destination. If it is, looping is involved and the message is ignored.
- ▪ **Policy routing:**

When a router receives a message, it can check the path. If one of the autonomous systems listed in the path is against its policy, it can ignore that path and that destination.

- **Optimum path:**
  The optimum path is the path that fits the organization like smaller number of autonomous systems.
  Other criteria, such as security, safety, and reliability, can also be applied.

BGP

- Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing.

*Types **of Autonomous** Systems:*

We can divide autonomous systems into three categories: stub, multihomed, and transit.

- **Stub AS**: A stub AS has only one connection to another AS. Data traffic, however, cannot pass through a stub AS. A stub AS is either a source or a sink. A good example of a stub AS is a small corporation or a small local ISP.
- **Multihomed AS**: A multihomed AS has more than one connection to other ASs, but it is still only a source or sink for data traffic. It does not allow transient traffic. A good example of a multihomed AS is a large corporation that is connected to more than one regional or national AS that does not allow transient traffic.
- **Transit AS**: A transit AS is a multihomed AS that also allows transient traffic. Good examples of transit ASs are national and international ISPs.

Path Attributes

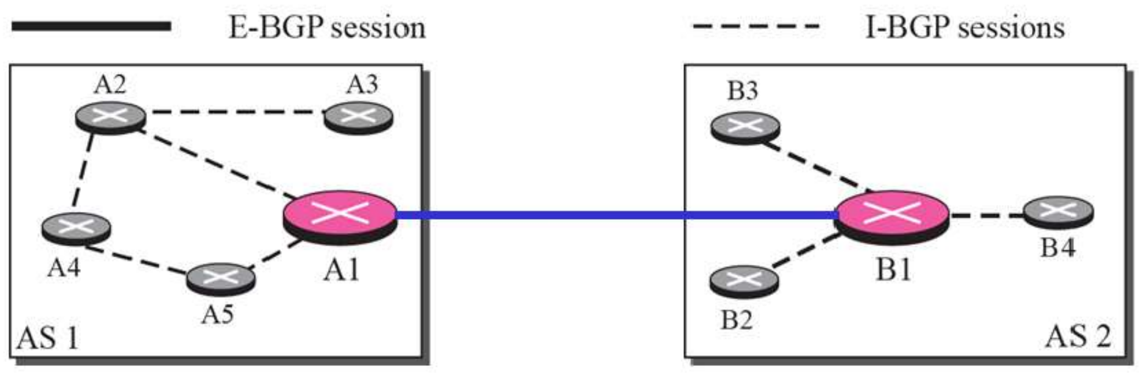
- The list of attributes helps the receiving router make a more-informed decision when applying its policy.
- Attributes are divided into two broad categories: well known and optional. A well-known Attribute is one that every BGP router must recognize. An optional attribute is one that needs not be recognized by every router.
- Well-known attributes are themselves divided into two categories: mandatory and discretionary. A *well-known mandatory attribute* is one that must appear in the description of a route. Ex: ORIGIN, AS_PATH. A *well-known discretionary attribute* is one that must be recognized by each router, but is not required to be included in every update message.
- The optional attributes can also be subdivided into two categories: transitive and non transitive. An *optional transitive attribute* is one that must be passed to the next router by the router that has not implemented this attribute. An *optional nontransitive attribute* is one that must be discarded if the receiving router has not implemented it.

BGP Sessions

- A session is a connection that is established between two BGP routers only for the sake of exchanging routing information.
- A session at the BGP level, as an application program, is a connection at the TCP level.
- BGP sessions are sometimes referred to as *semi permanent connections.*

External and Internal BGP:

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



- The session established between AS1 and AS2 is an E-BOP session.
- Two speaker routers need to collect information from other routers in the autonomous systems. This is done using I-BOP sessions.

23. WORLD WIDE WEB (WWW)

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.

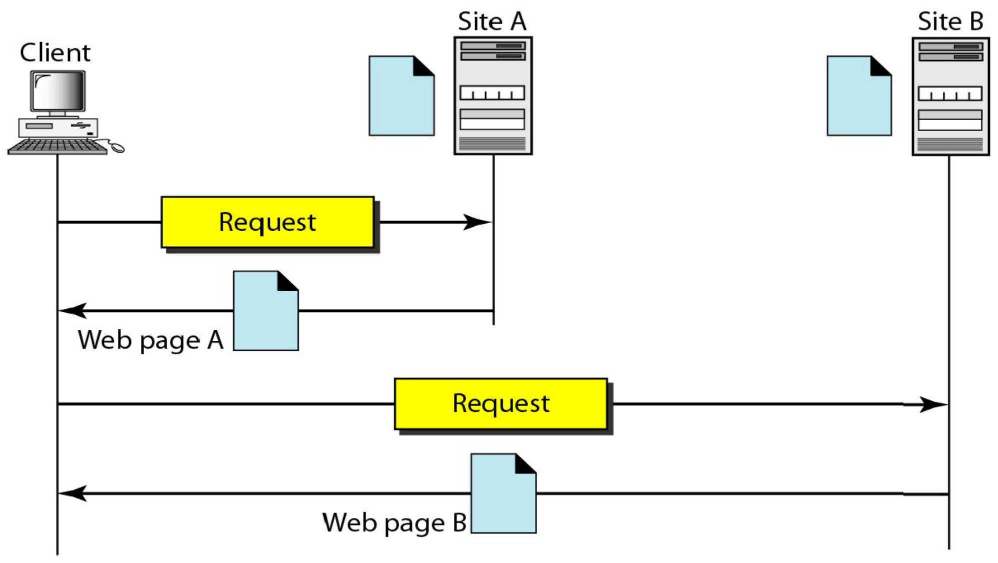
Topics discussed in this section:

- Architecture
- Client (Browser)
- Server
- Uniform resource locator
- Cookies
- Web documents

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 can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers.
In below figure, the client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL (uniform resource locator).

Architecture of WWW



Client (Browser):

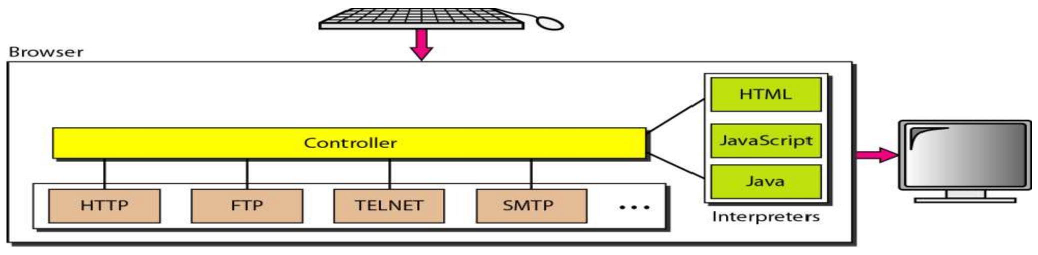
Each browser usually consists of three parts: a controller, client protocol, and interpreters.

Controller: 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.

Client protocol: The client protocol can be one of the protocols such as FTP or HTIP or SMTP or TELNET…etc.

Interpreter: The interpreter can be HTML, Java, or JavaScript, depending on the type of document.

Browser

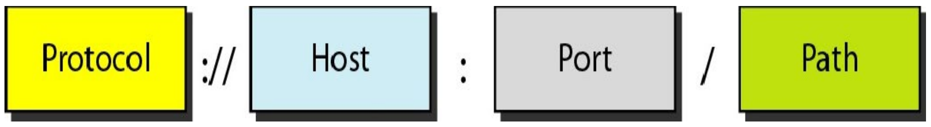


Server:

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

Uniform Resource Locator:

The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path in the given figure



Protocol: The *protocol* is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common t day is HTTP.

Host: The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www".

Port: 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: Path is the pathname of the file where the information is located.

COOKIES:

- The WWW was originally designed as a stateless entity.
- Cookies are needed for extending functionalities of the Web, such as to remember past client in order to show a customized webpage

Creation and storage of cookies:

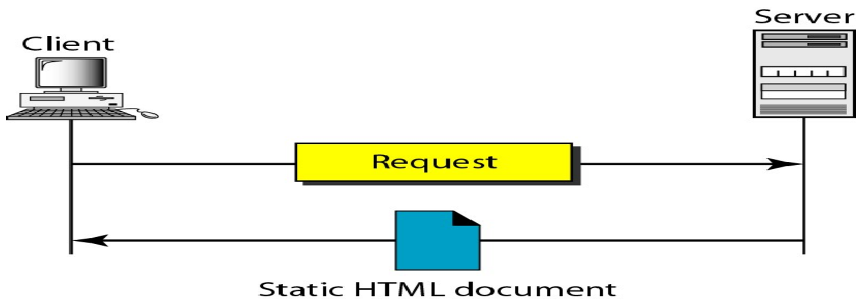
- When a server receives a request from a client, it stores information about the client in a file or a string.
- The server includes the cookie in the response that it sends to the client.
- When the client receives the response, the browser stores the cookie in the cookie directory

WEB DOCUMENTS :

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

Static Documents: Static documents are fixed-content documents that are created and stored in a server.The client can get only a copy of the document. In other words, the contents of the fileare determined when the file is created, not when it is used.

Static Document



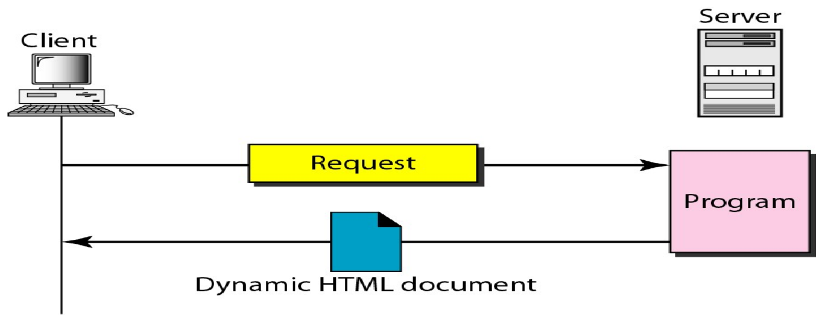
The language used for creating web pages is**Hypertext Markup Language (HTML).**

Dynamic Documents:A dynamic documentis 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.

Common Gateway Interface (CGI)

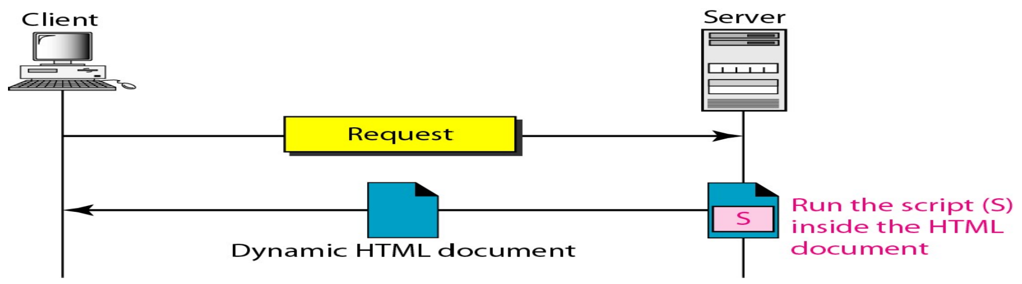
CGl is not a new language; instead, it allows programmers to use any of several languages such as C, C++, Bourne Shell, Korn Shell, C Shell, Tcl, or Perl.

Dynamic document using CGI



A few technologies have been involved in creating dynamic documents using scripts. Among the most common are Hypertext Preprocessor (PHP), which uses the Perl language; Java Server Pages (JSP), which uses the Java language for scripting; Active Server Pages (ASP), a Microsoft product which uses Visual Basic language for scripting; and ColdFusion, which embeds SQL database queries in the HTML document. Dynamic documents are sometimes referred to as server-site dynamic documents.

Dynamic document using server-site script

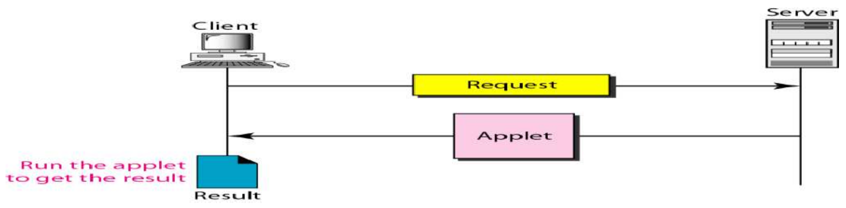


Active Documents:

For many applications, we need a program or a script to be run at the client site. These
are called active documents.

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.

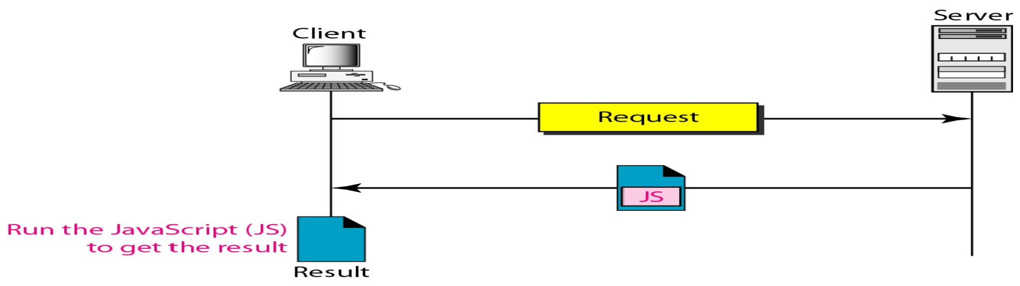
Active document using Java applet



An applet is a program written in Java on the server. It is compiled and ready to be
run. The document is in byte-code (binary) format.

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.

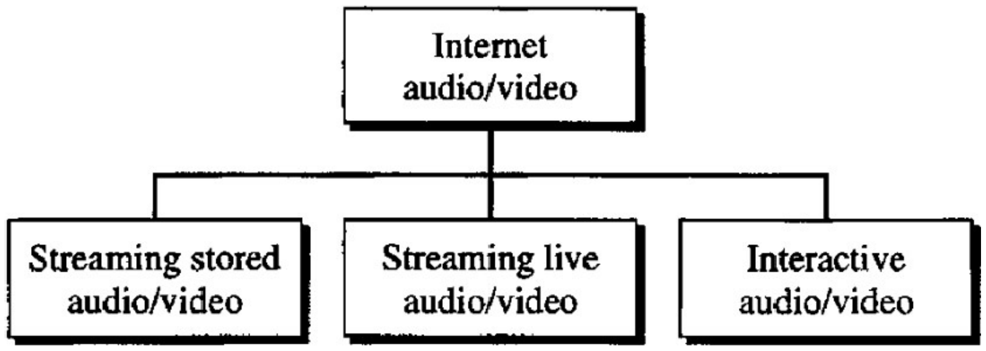
Active document using client-site script



Active documents are sometimes referred to as client-site dynamic documents.

24. Explain about RTP,RTCP, Streaming live audio and video?

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



Now in this we are going to discuss about Streaming live audio and video

Streaming live audio and video:

"*Streaming live audio/video refers to the broadcasting of radio and TV programs 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.

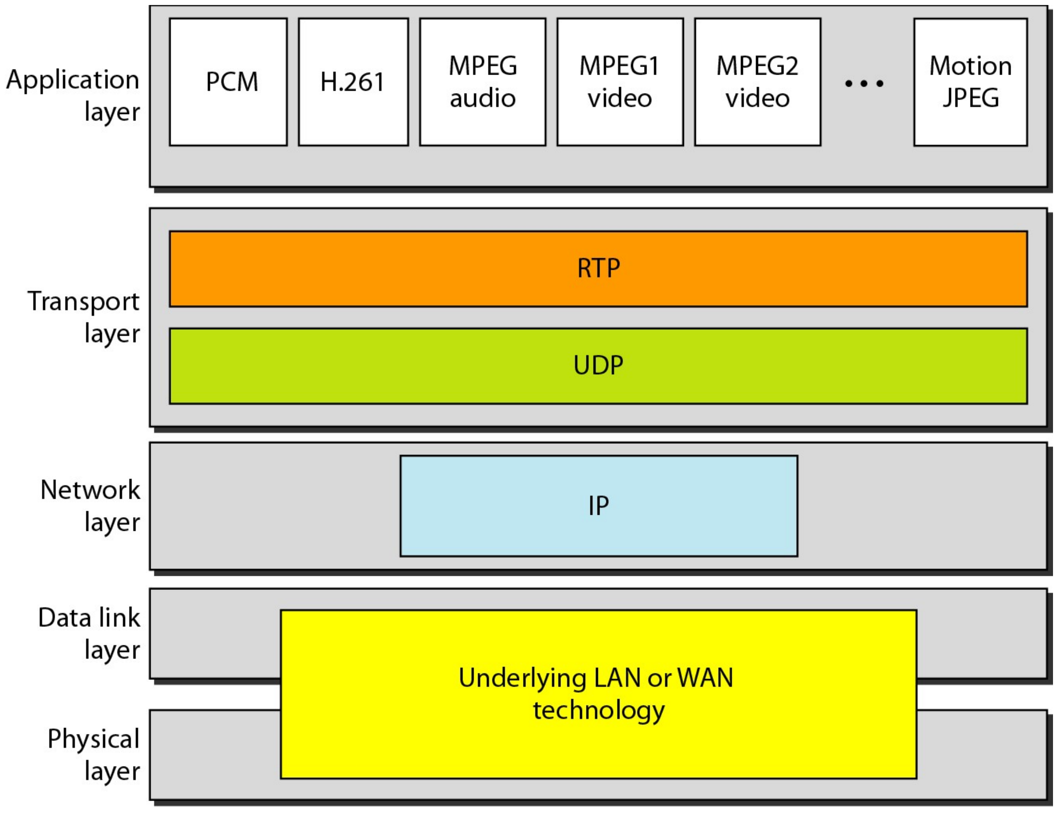
(Or)

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. 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.

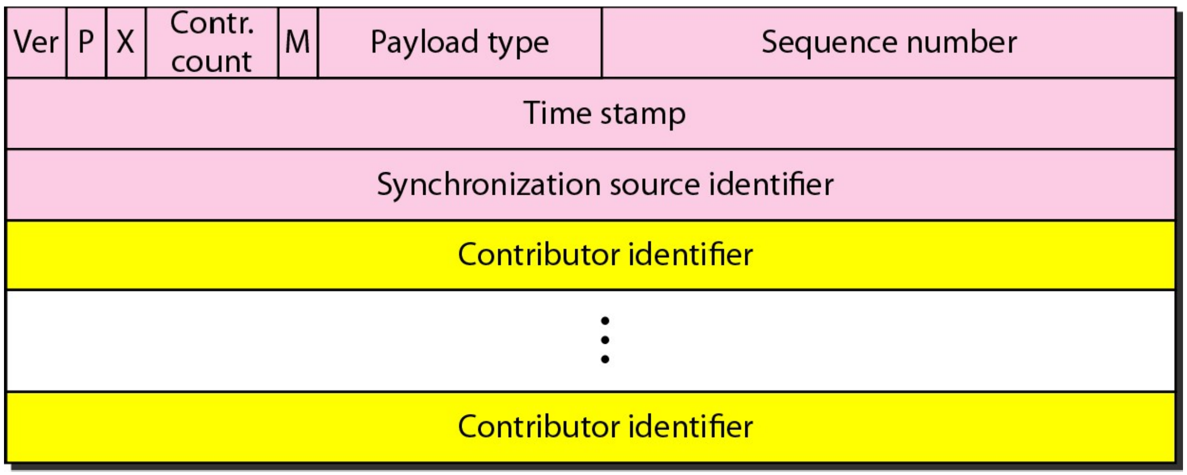
RTP (Real-time Transport Protocol):

Real-time Transport Protocol (RTP) is the protocol designed to handle real-time traffic on the Internet. RTP does not have a delivery mechanism (multicasting, port numbers, and so on); it must be used with UDP. RTP stands between UDP and the application program. The main contributions of RTP are time-stamping, sequencing, and mixing facilities. Figure below shows the position of RTP in the protocol suite.



RTP

RTP Packet Format:

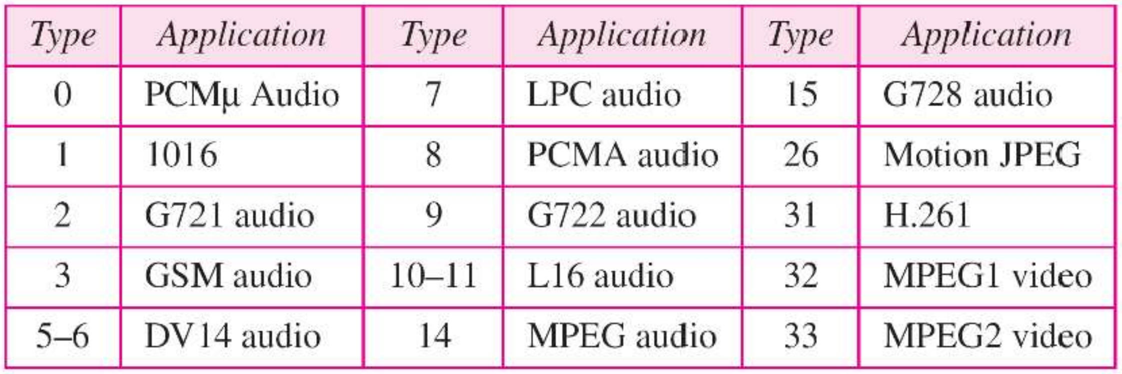


A description of each field follows:

- **Ver :** This 2-bit field defines the version number. The current version is 2.
- **P** : This I-bit field, if set to 1, indicates the presence of padding at the end of the packet. In this case, the value of the last byte in the padding defines the length of the padding. Padding is the norm if a packet is encrypted. There is no padding if the value of the P field is O.
- **X:** This I-bit field, if set to 1, indicates an extra extension header between the basic header and the data. There is no extra extension header if the value of this field is O.
- **Contributor count:**This 4-bit field indicates the number of contributors. Note that we can have a maximum of 15 contributors because a 4-bit field only allows a number between 0 and 15.

- **M:** This I-bit field is a marker used by the application to indicate, for example, the end of its data.

Payload type: This 7-bit field indicates the type of the payload. Several payload types have been defined so far. We list some common applications in Table 29.1.



Type	*Application*	*Type*	*Application*	*Type*	*Application*
0	PCMμ Audio	7	LPC audio	15	G728 audio
1	1016	8	PCMA audio	26	Motion JPEG
2	G721 audio	9	G722 audio	31	H.261
3	GSM audio	10–11	L16 audio	32	MPEG1 video
5–6	DV14 audio	14	MPEG audio	33	MPEG2 video

Payload types

- **Sequence number:**This field is 16 bits in length. It is used to number the RTP packets. The sequence number of the first packet is chosen randomly; it is incremented by 1 for each subsequent packet. The sequence number is used by the receiver to detect lost or out-of-order packets.
- **Timestamp:**This is a 32-bit field that indicates the time relationship between packets. The timestamp for the first packet is a random number. For each succeeding packet, the value is the sum of the preceding timestamp plus the time the first byte is produced (sampled). The value of the clock tick depends on the application. For example, audio applications normally generate chunks of 160 bytes; the clock tick for this application is 160. The timestamp for this application increases 160 for each RTP packet.
- **Synchronization source identifier:** If there is only one source, this 32-bit field defines the source. However, if there are several sources, the mixer is the synchronization source and the other sources are contributors. The value of the source identifier is a random number chosen by the source. The protocol provides a strategy in case of conflict (two sources start with the same sequence number).
- **Contributor identifier:**Each of these 32-bit identifiers (a maximum of 15) defines a source. When there is more than one source in a session, the mixer is the synchronization source and the remaining sources are the contributors.

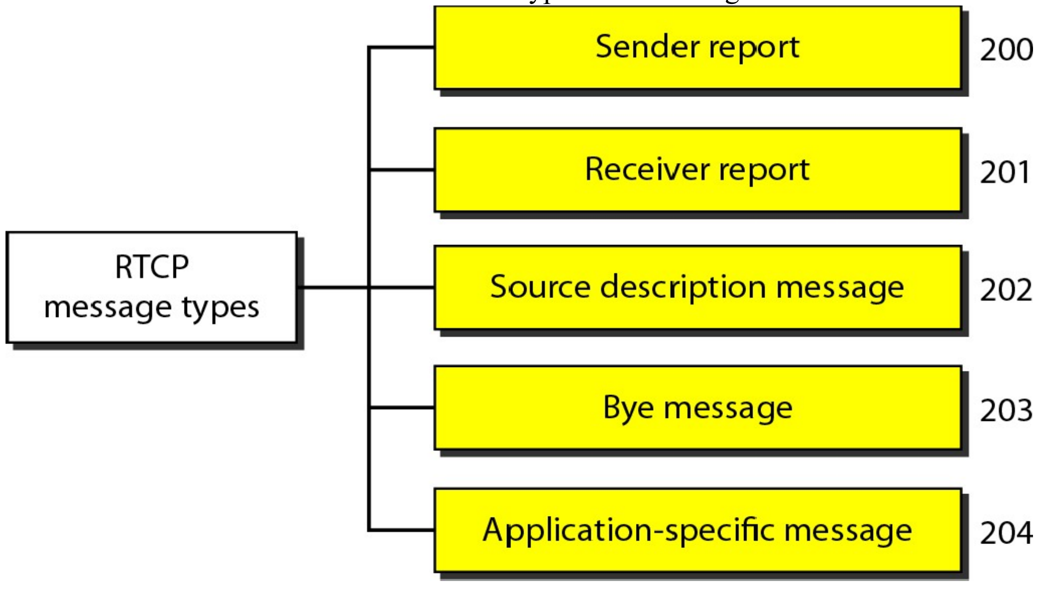
UDP Port: Although RTP is itself a transport layer protocol, the RTP packet is not encapsulated directly in an IP datagram. Instead, RTP is treated as an application program and is encapsulated in a UDP user datagram. However, unlike other application programs, no well-known port is assigned to RTP. The port can be selected on demand with only one restriction: The port number must be an even number. The next number (an odd number) is used by the companion of RTP, Real-time Transport Control Protocol (RTCP).

“RTP uses a temporary even-numbered udp port”.

RTCP

RTP allows only one type of message, one that carries data from the source to the destination.
In many cases, there is a need for other messages in a session. These messages control the flow and quality of data and allow the recipient to send feedback to the source or sources. Real-time Transport Control

Protocol (RTCP) is a protocol designed for this purpose. RTCP has five types of messages, as shown in Figure below. The number next to each box defines the type of the message.



RTCP message types

Sender Report:

The sender report is sent periodically by the active senders in a conference to report transmission and reception statistics for all RTP packets sent during the interval. The sender report includes an absolute timestamp, which is the number of seconds elapsed since midnight on January 1, 1970. The absolute timestamp allows the receiver to synchronize different RTP messages. It is particularly important when both audio and video are transmitted (audio and video transmissions use separate relative timestamps).

Receiver Report:

The receiver report is for passive participants, those that do not send RTP packets. The report informs the sender and other receivers about the quality of service.

Source Description Message:

The source periodically sends a source description message to give additional information about itself. This information can be the name, e-mail address, telephone number, and address of the owner or controller of the source.

Bye Message:

A source sends a bye message to shut down a stream. It allows the source to announce that it is leaving the conference. Although other sources can detect the absence of a source, this message is a direct announcement. It is also very useful to a mixer.

Application-Specific Message:

The application-specific message is a packet for an application that wants to use new applications (not defined in the standard). It allows the definition of a new message type.

UDP Port:

RTCP, like RTP, does not use a well-known UDP port. It uses a temporary port. The
UDP port chosen must be the number immediately following the UDP port selected for RTP. It must be an odd-numbered port.

"RTCP uses an odd-numbered UDP port number that follows the port number selected for RTP."

25. Explain the following ?

1. Domain name space.

2. Resolution.

Ans:

Domain name space:-

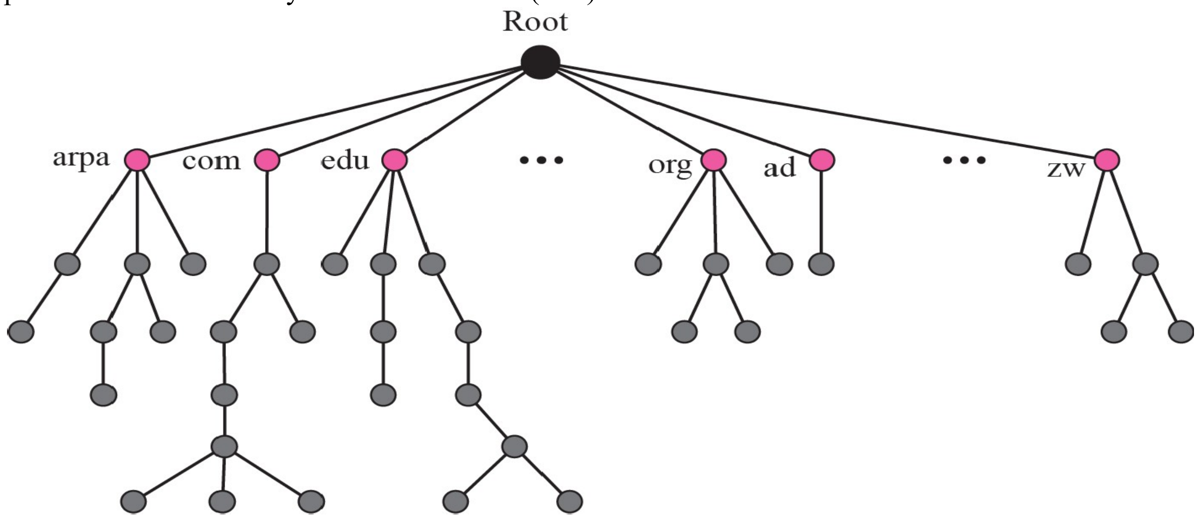
To identify an entity ,Tcp /Ip uses the IP addresses which uniquely identifies the connection of a host to the internet.
But people use names instead of addresses for ex: to send a mail we type mail id of that particular person instead of addresses. So ,we need a unique name spaces for every machine.

Name spaces can be organized in two ways:

---- Flat name space: A name in this name space is a sequence of characters without any structure.

----Hierarchical name space: In a hierarchical name space, each name is made of several parts. The first part can
define the nature of the organization, the second part can define the name of an organization ,the third part can define departments in the organization.

Domain name space is the design in which names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.



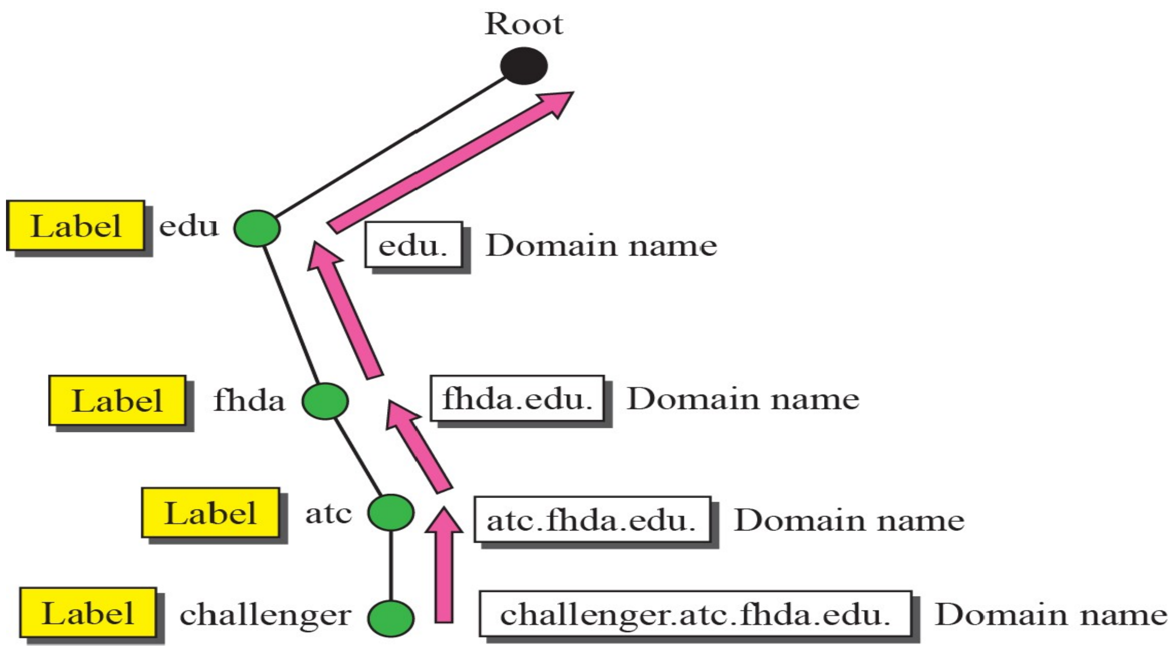
This consists of

a) label
b) Domain name

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).
The children of the node must also be a different label to guarantee uniqueness.

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).

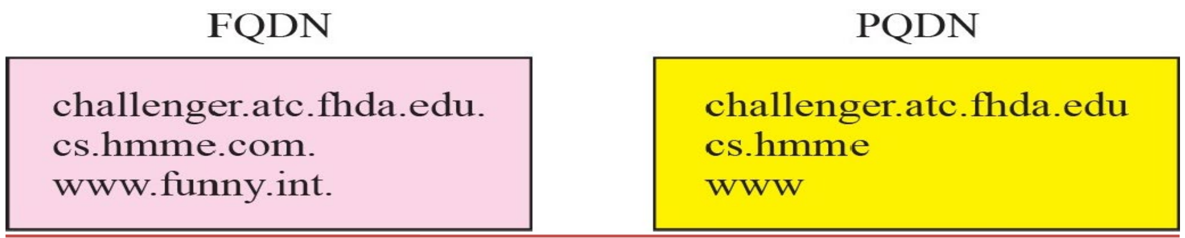


There are two types of domain names:

1. Fully Qualified names
2. Partially qualified names

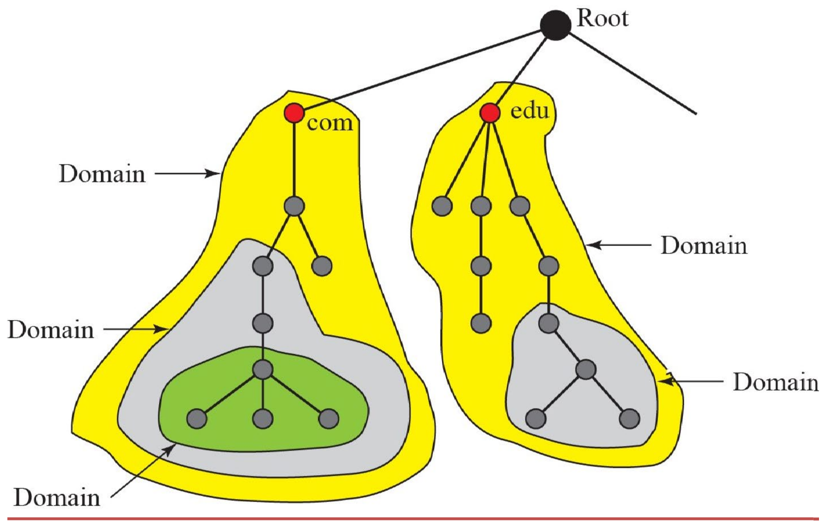
FULLY QUALIFIED NAMES:
If a label is terminated by a null string, it is called a fully qualified domain name(FQDN).
An FQDN is a domain name that contains the full name of a host.
For example, the domain name challenger.ate.tbda.edu.

PARTIALLY QUALIFIED NAMES:
If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root.



FQDN	PQDN
challenger.atc.fhda.edu.	challenger.atc.fhda.edu
cs.hmme.com.	cs.hmme
www.funny.int.	www

Domain:
A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree.



RESOLUTION:

Mapping a name to an address or an address to a name is called *name-address resolution.*

RESOLVER:

A host that needs to map an address to a
name or a name to an address calls a DNS client called a resolver.

Mapping Names to addresses: The resolver gives a domain name to the server and asks for the corresponding address**.**

Mapping addresses to names: A client can send an IP address to a server to be mapped to a domain name.

This is called a PTR query.

To answer queries of this kind, DNS uses the inverse domain.

There are two types of resolutions

---RECURSIVE RESOLUTION

---ITERATIVE RESOLUTION

Recursive resolution:

If a resolver asks for a recursive answer from a name server, resolver expects a final answer from that server.

If the server is the authority for the domain name, it checks the database and responds.

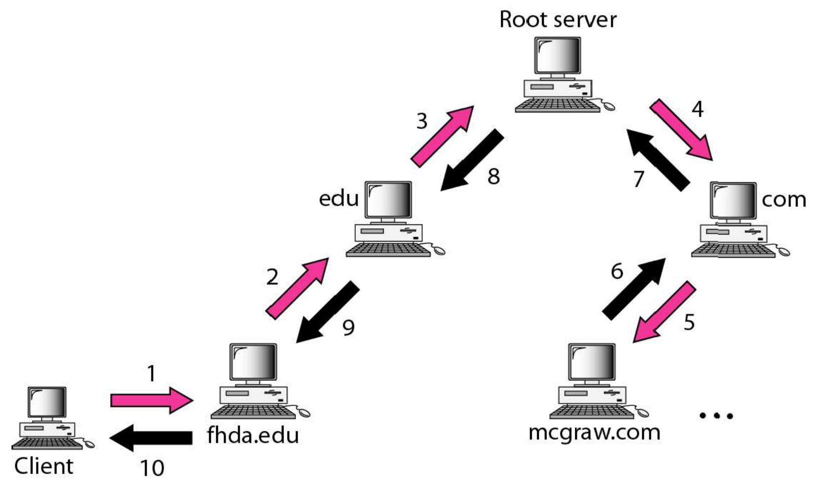
If the server is not the authority, it sends the request to another server.

If that server is the authority, it responds.

If not, it forwards the request on.

Eventually the authoritative server is reached and responds.

The resolved name travels back to the requesting client.



Iterative resolution:

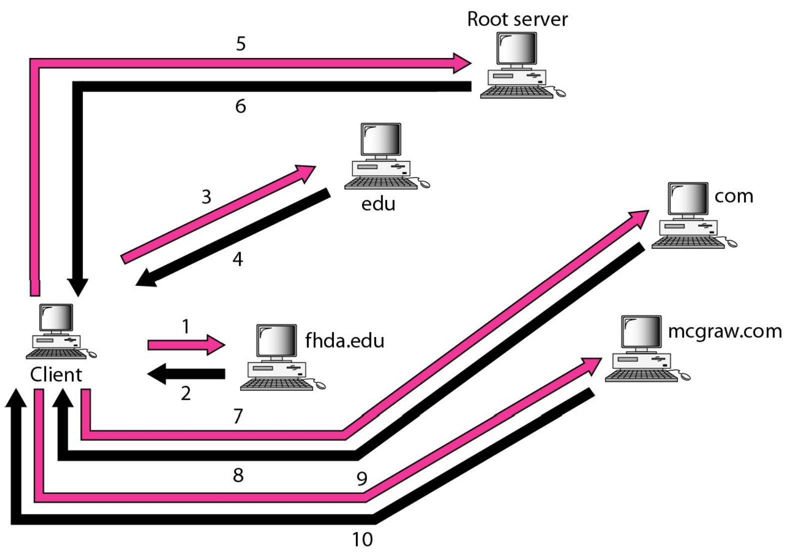
If a client does not ask for a recursive answer, the mapping can be done iteratively.

If the first DNS server is not the authority, it sends the Client IP address of the server that it thinks is the Authority.

The client then sends a request to that server.

If it is not the authority, it refers the client to another Server.

The process repeats until the authoritative server is reached and the name is resolved.



PROGRAM 1

AIM: ***WRITE A JAVA PROGRAM TO IMPLEMENT BIT STUFFING***

DESCRIPTION:

- To prevent *data* being interpreted as *control* information. For example, many frame-based protocols, such as X.25, signal the beginning and end of a frame with six consecutive 1 bits. Therefore, if the actual data being transmitted has six 1 bits in a row, a zero is inserted after the first 5 so that the dat is not interpreted as a frame delimiter. Of course, on the receiving end, the stuffed bits must be discarded.

- For protocols that require a fixed-size frame, bits are sometimes inserted to make the frame size equal to this set size.

- For protocols that required a continuous stream of data, zero bits are sometimes inserted to ensure that the stream is not broken.

SOURCE CODE:

```
import java.util.*;

import java.lang.*;

import java.io.*;

public class Bitstuffing

{

public static void main(String args[])throws IOException

{

System.out.println("enter the binary message");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String  str=br.readLine();

String  res=new  String();

int count=0;

for(int i=0;i<str.length();i++)

{

System.out.println(str.charAt(i));
if(str.charAt(i)!='1'&&str.charAt(i)!='0')

{

System.out.println("enter the binary values");

return;

}

if(str.charAt(i)=='1')

{

count++;

res=res+str.charAt(i);

}

else

{

res=res+str.charAt(i);

count=0;

}

if(count==5)

{

res=res+'0';

count=0;

}

}

System.out.println("the original message:"+str);

System.out.println("the Bitstuffing message:"+res);

res="01111110"+res+"01111110";

System.out.println("the message stuffing:"+res);
```

```
}

}
```

<u>Output 1:-</u>

```
enter the binary message

011111101

0

1

1

1

1

1

1

0

1

the original message:011111101

the Bitstuffing message:0111110101

the message stuffing:01111110011111010101111110
```

<u>Output 2:-</u>

```
enter the binary message

10110

1

0

1

1

0
```

```
the original message:10110

the Bitstuffing message:10110

the message stuffing:01111110101100111111O
```

PROGRAM 2

AIM: ***WRITE A JAVA PROGRAM TO IMPLEMENT BIT DESTUFFING PROGRAM***

DESCRIPTION:One the bits are stuffed and sent by the sender the receiver receives the msg.The receiver has to decode it. The algorithm used is destuffin algorithm. It removes a bit when consequtive 5 ones appear.

1. Input the stuffed sequence.2. Remove start of frame from sequence.3. For every bit in input,a. Append bit to output sequence. b. Is bit a 1?Yes: Increment count. If count is 5, remove next bit (which is0) and reset count. No: Set count to 0.4. Remove end of frame bits from sequence.

SOURCE CODE:

```
import java.util.*;

import java.lang.*;

import java.io.*;

public class Bitunstuffing

{

public static void main(String args[])throws IOException

{

System.out.println("enter the binary message");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str=br.readLine();

String res=new String();

int count=0;

for(int i=8;i<str.length()-8;i++)

{

System.out.println(str.charAt(i));

if(str.charAt(i)!='1'&&str.charAt(i)!='0')

{
```

```
System.out.println("enter the binary values");

return;

}

if(str.charAt(i)=='1')

{

count++;

res=res+str.charAt(i);

}

else

{

res=res+str.charAt(i);

count=0;

}

if(count==5)

{

i++;

count=0;

}

}

System.out.println("the string is:"+res);

}

}
```

Output 1:-

```
enter the binary message

01111110011111010101111110
```

```
0
1
1
1
1
1
1
0
1
the string is:011111101
```

<u>Output 2:-</u>

```
enter the binary message
011111101011001111110
1
0
1
1
0
the string is:10110
```

PROGRAM 3

AIM: ***WRITE A JAVA PROGRAM TO IMPLEMENT CHARACTER STUFFING***

DESCRIPTION: In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

SOURCE CODE:

```
import java.io.*;

import java.util.*;

import java.lang.*;

public class Bytestuffing

{

public static void main(String args[])throws IOException

{

System.out.println("enter message");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str=br.readLine();

String res=new String();

System.out.println("enter starting flag");

char sf=br.readLine().charAt(0);

System.out.println("enter ending flag");

char ef=br.readLine().charAt(0);

for(int i=0;i<str.length();i++)

{

System.out.println("\n");
System.out.println(str.charAt(i));

if(str.charAt(i)==sf)

{

res=res+str.charAt(i)+sf;

}

else

if(str.charAt(i)==ef)

{

res=res+str.charAt(i)+ef;

}

else

{

res=res+str.charAt(i);

}

}

System.out.println("the original messsage is:\n" + str);

System.out.println("the stuffed message is:\n"+res);

res=sf+res+ef;

System.out.println("the message after stuffing :\n"+res);

}

}
```

Output 1:-

```
enter message

cnoslab

enter starting flag
```

```
a

enter ending flag

b


c

n

o

s

l

a

b

the original messsage is:

cnoslab

the stuffed message is:

cnoslaabb

the message after stuffing:

acnoslaabbb
```

<u>Output2:-</u>

```
enter message

aceec

enter starting flag

a

enter ending flag

e
```

```
a
c
e
e
c
the original messsage is:
aceec
the stuffed message is:
aacceecc
the message after stuffing:
aaacceeccc
```

PROGRAM 4

AIM: ***A JAVA PROGRAM TO IMPLEMENT CHARACTER DESTUFFING***
DESCRIPTION:

1. Input stuffed sequence.
2. Remove start of frame from sequence (DLE STX).
3. For every character in input,
    a. Append character to output sequence.
    b. Is character DLE?Yes: Remove next DLE from input sequence. No: Continue.
4. Remove stop of frame character to output sequence (DLE ETX).

SOURCE CODE:

```
import java.io.*;

import java.util.*;

import java.lang.*;

public class Byteunstuffing

{

public static void main(String args[])throws IOException

{

System.out.println("enter message");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str=br.readLine();

String res=new String();

char sf=str.charAt(0);

char ef=str.charAt(str.length()-1);

for(int i=1;i<str.length()-1;i++)

{

if(str.charAt(i)==sf)

{

++i;

res=res+str.charAt(i);
}
else if(str.charAt(i)==ef)
{
++i;
res=res+str.charAt(i);
}
else
res=res+str.charAt(i);
}
System.out.println("the string is:\n"+res);
}
}
```

Output1:-

```
enter message
acnoslaabbb
the string is:
cnoslab
```

Output2:-

```
enter message
aaacceeccc
the string is:
aceec
```

PROGRAM 5

AIM:***WRITE A JAVA PROGRAM TO IMPLEMENT CRC CODE***

DESCRIPTION:

1. Let r be the degree of G(x). Append r zero bits to the low-order end of theframe, so it now contains m+r bits and corresponds to the polynomial x^r M(x).
2. Divide the bit string corresponding to G(X) into the bit string corresponding to x^r M(x) using modulo-2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit stringcorresponding to x^r M(x) using modulo-2 subtraction. The result s the checksummed frame to be transmitted. Call its polynomial T(x).The CRC-CCITT polynomial is x^16+ x^12+ x^5+ 1

SOURCE CODE:

```
import java.lang.*;

import java.util.*;

import java.io.*;

class CRC

{

public static void main(String args[]) throws IOException

{

CRCFN obj=new CRCFN();

obj.send();

obj.receive();

}

}

class CRCFN

{

int n,h,l,dl;

char p[]=new char[50];

char t[]=new char[50];

char om[]=new char[50];

char dv[]=new char[50];

char r[]=new char[50];
```

```
int i,j,k,z;

String str1=new String("");

String str2=new String("");

String poly=new String("");

String tempdiv=new String("");

void send() throws IOException

{

BufferedReader br =new BufferedReader(new InputStreamReader(System.in));

System.out.println("enter crc to be used");

System.out.println("enter 12 for crc -12 \n 160 for crc 16\n 161 for crc-16-cit\n");

n=Integer.parseInt(br.readLine());

if(!((n==12||(n==160)||(n==161))))

{

System.out.println("invalid");

System.exit(0);

}

System.out.println("\nat sender side");

System.out.println(" --------------- ");

System.out.println("\n enter the original frame");

str1=br.readLine();

om=str1.toCharArray();

for(int i=0;i<om.length;i++)

System.out.print(om[i]);

h=om.length;

str2+=str1;
```

```
System.out.println("\n lenght and original message\n"+h+"\t\n"+str2);

if(n==12)

{

poly+=("1100000001111");

tempdiv+=("0000000000000");

str2+=("000000000000");

System.out.println(poly+"   \n"+tempdiv+"     \n" + str2);

}

else if(n==160)

{

poly+=("11000000000000101");

tempdiv+=("00000000000000000");

str2+=("000000000000");

System.out.println(poly+"   \n"+tempdiv+"       \n" +str2);

}

else if(n==161)

{

poly+=("1000100000010001");

tempdiv+=("00000000000000000");

str2+=("0000000000000000");

System.out.println(poly+"   \n" +tempdiv+"   \n"+str2);

}

else

{

System.out.println("invalid");
System.exit(0);

}

dv=str2.toCharArray();

p=poly.toCharArray();

t=tempdiv.toCharArray();

l=p.length;

dl=dv.length;

System.out.println("\nlenght of polynamial and dividend\n"+l+"\t\n  "+dl);

System.out.println("\naugmented didvdend\n");

for(int i=0;i<dv.length;i++)

System.out.print(dv[i]);

System.out.println("\n");

System.out.println("\ngenerator\n");

for(int i=0;i<p.length;i++)

System.out.print(p[i]);

System.out.println("\n");

System.out.println("\ntemp data\n");

for(int i=0;i<t.length;i++)

System.out.print(t[i]);

System.out.println("\nthe divisor for each step\nof modulo-2");

for(i=0;i<h;i++)

{

if(dv[i]=='0')

{

for(j=i,k=0;j<(l+i)&&k<l;j++,k++)
{
if(dv[j]==t[k])
dv[j]='0';
else
dv[j]='1';
}
}
else if(dv[i]=='1')
{
for(j=i,k=0;j<(l+i)&&k<l;j++,k++)
{
if(dv[j]==p[k])
dv[j]='0';
else
dv[j]='1';
}
}
else if((dv[i]!='0')||(dv[i]!='1'))
{
System.out.println("invalid frame");
System.exit(0);
}
for(z=i;z<(l+i);z++)
System.out.print(dv[z]);
System.out.println("\n");
```

```
}

for(i=h,j=0;i<=(dl-1);i++,j++)

r[j]=dv[i];

String rstr=new String(r);

System.out.println("\nthe reminder is\n"+rstr);

str1+=rstr;

System.out.println("\n the frame sent is:\n"+str1);

}

void receive()throws IOException

{

//char om[50],dv[50],r[20];

char om[]=new char[50];

char dv[]=new char[50];

char r[]=new char[50];

int i,j,k,z,c=0;

String omstr=new String("");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("\n at the receiver\n");

System.out.println("...................\n");

System.out.println("\n enter the frame received:\n");

omstr=br.readLine();

om=omstr.toCharArray();

for(i=0;i<om.length;i++)

dv[i]=om[i];

//strcpy(dv,om);
```

```
System.out.println("\n the divisor for each step \n of modulo-2 division is:\n");

for(i=0;i<=h-1;i++)

{

if(dv[i]=='0')

{

for(j=i,k=0;j<(l+i)&&k<l;j++,k++)

{

if(dv[j]==t[k])

dv[j]='0';

else

dv[j]='1';

}

}

else if(dv[i]=='1')

{

for(j=i,k=0;j<(l+i)&&k<l;j++,k++)

{

if(dv[j]==p[k])

dv[j]='0';

else

dv[j]='1';

}

}

else

{
```

```
System.out.println("invalid frame\n");

System.exit(0);

}

for(z=i;z<13+i;z++)

System.out.print(dv[z]);

System.out.println("\n");

}

for(i=h,j=0;i<=(dl-1);i++,j++)

{

r[j]=dv[i];

if(r[j]=='0')

c++;

}

//r[j]='\0';

System.out.println("\n the remainder is\n");

for(int q=0;q<r.length;q++)

System.out.print(r[q]);

System.out.println("\n\n\n"+c);

if(c==l-1)

{

System.out.println("\n the frame received is error free:\n");

//om[h]='\0';

System.out.println("\n the original frame is \n");

for(int q=0;q<om.length;q++)

System.out.print(om[q]);
System.out.println("\n the original message is \n");

for(int x=0;x<h;x++)

System.out.print(om[x]);

}

else

System.out.println("\n received frame is corrupted:FRAME HAVE ERRORS\n");

}

}
```

Output1:-

```
enter crc to be used

 enter 12 for crc-12
 160 for crc-16
 161 for crc-16-ccitt
12

 at the sender side
........................

 enter the original frame
10110
10110
Length and original message
5
10110
1100000001111
0000000000000
10110000000000000
length of polynomial and dividend
13
17
augmented didvdend
10110000000000000

Generator
1100000001111
```

```
temp data
00000000000000

the divisor for each step
of modulo-2
0111000001111
0010000010001
0100000100010
0100001001011
010001011001

 the remainder is
100010011001

 the frame sent is:
10110100010011001

 at the receiver
........................

 enter the frame received:
10110100010011001

 the divisor for each step
of modulo-2 division is:
0111010000110
0010100000010
0101000000100
0110000000111
0000000000000

 the remainder is 0000000000000
12
the frame received is error free:

the original frame is
10110100010011001

the original message is
10110
```

Output2:-

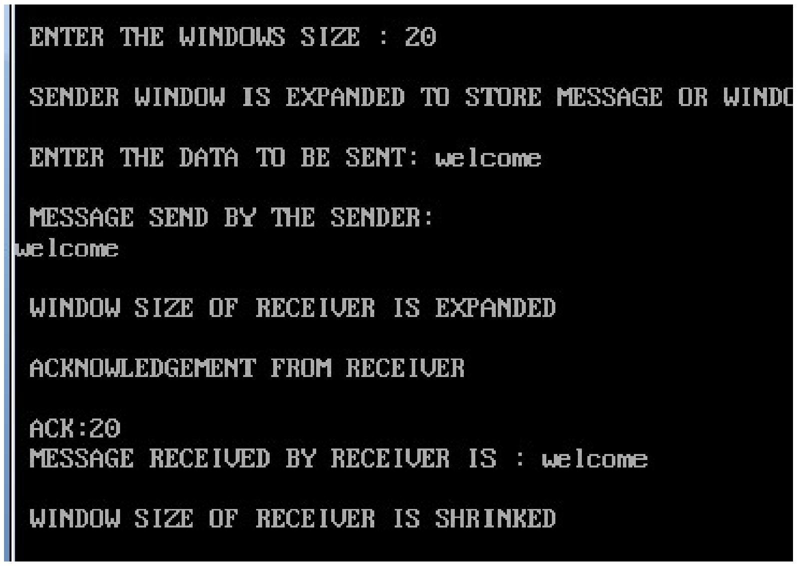
```
enter crc to be used
enter 12 for crc-12
 160 for crc-16
 161 for crc-16-ccitt
32
Invalid
```

PROGRAM 6

AIM:***DEVELOP A SIMPLE DATA LINK LAYER THAT PERFORMS THE FLOW CONTROL USING THE SLIDING WINDOW PROTOCOL, AND LOSS RECOVERY USING THE GO-BACK-N MECHANISM.***

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char sender[50],receiver[50];
int i,winsize;
 printf("\n ENTER THE WINDOWS SIZE : ");
scanf("%d",&winsize);
 printf("\n SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR WINDOW \n");
 printf("\n ENTER THE DATA TO BE SENT: ");
fflush(stdin);
gets(sender);
for(i=0;i<winsize;i++)
receiver[i]=sender[i];
receiver[i]=NULL;
 printf("\n MESSAGE SEND BY THE SENDER:\n");
 puts(sender);
 printf("\n WINDOW SIZE OF RECEIVER IS EXPANDED\n");
 printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");
for(i=0;i<winsize;i++);
printf("\n ACK:%d",i);
 printf("\n MESSAGE RECEIVED BY RECEIVER IS : ");
 puts(receiver);
 printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");
getch();
}
```



```
 ENTER THE WINDOWS SIZE : 20

 SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR WINDO

 ENTER THE DATA TO BE SENT: welcome

 MESSAGE SEND BY THE SENDER:
welcome

 WINDOW SIZE OF RECEIVER IS EXPANDED

 ACKNOWLEDGEMENT FROM RECEIVER

 ACK:20
 MESSAGE RECEIVED BY RECEIVER IS : welcome

 WINDOW SIZE OF RECEIVER IS SHRINKED
```

PROGRAM 7 :

AIM:*WRITE A JAVA PROGRAM TO IMPLEMENT DIJKSTRA'S ALGORITHM TO COMPUTE THE SHORTEST PATH THROUGH A GRAPH.*

DESCRIPTION: Let the node at which we are starting be called the **initial node**. Let the **distance of node Y** be the distance from the **initial node** to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the *unvisited set* consisting of all the nodes except the initial node.
3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances. For example, if the current node A is marked with a tentative distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbor has been examined, it is not marked as "visited" at this time, and it remains in the *unvisited set*.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again; its distance recorded now is final and minimal.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal), then stop. The algorithm has finished.
6. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step

SOURCE CODE:

```
Import  java.io.*;

Import  java.lang.*;

Import java.util.*;

class STATES

{

Int l;

Int p;

Int label;

}

//state [10];

class DIJ
```

```
{

public static void main(String args[]) throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int INF=1000;

int i,n,k,t,s,j,dest,temp1,min;

int path[]=new int[10];

int x,y;

int route[]=new int[10];

int dist[][]=new int[10][10];

STATES state[]=new STATES[10];

for(i=0;i<10;i++)

state[i] = new STATES();

System.out.println("Enter no.of nodes");

n=Integer.parseInt(br.readLine());

for(i=1;i<=n;i++)

{

System.out.println("enter the no.of neighbours for node ::"+i);

j=Integer.parseInt(br.readLine());

System.out.println("enter Neighbour number ,Distance");

for(k=0;k<j;k++)

{

temp1=Integer.parseInt(br.readLine());

dest=Integer.parseInt(br.readLine());

dist[i][temp1]=dest;
}
}
System.out.println("\n\n\tmatrix\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(i==j && dist[i][j]==0)
System.out.print("\t "+dist[i][j]);
else if(i!=j && dist[i][j]==0)
System.out.print("\t "+99);
else
System.out.print("\t "+dist[i][j]);
}
System.out.println("\n");
}
System.out.println("To find the path \n\n enter starting node");
t=Integer.parseInt(br.readLine());

System.out.println("Enter ending node");
s=Integer.parseInt(br.readLine());
for(i=1;i<=n;i++)
{
state[i].p=-1;
state[i].l=INF;
```

```
state[i].label=0;
}
state[t].l=0;
state[t].label=1;
k=t;
do
{
for(i=1;i<=n;i++)
{
if(dist[k][i]!=0 && state[i].label==0)
{
if(state[k].l+dist[k][i]<state[i].l)
{
state[i].p=k;
state[i].l=state[k].l+dist[k][i];
}
}
}
k=0;
min=INF;
for(i=1;i<=n;i++)
{
if(state[i].label==0 && state[i].l<min)
{
min=state[i].l;
k=i;

}

}

state[k].label=1;

}while(k!=s);

i=0;

k=s;

System.out.println("\n Shortest Path is:");

x=0;

do

{

path[i]=k+1;

route[x]=k;

x++;

i++;

k=state[k].p;

}while(k>0);

for(y=x-1;y>=0;y--)

System.out.print(route[y]+"->");

System.out.println("\n Minimum cost is "+min);

}

}
```

<u>Output1:-</u>

```
enter no. of nodes

5
```

```
4

enter the no. of neighbours for node ::1

3

enter Neighbour number ,Distance

2

2

5

3

3

1

enter the no.of neighbours for node ::2

2

enter Neighbour number ,Distance

1

2

3

2

enter the no.of neighbours for node ::3

3

enter Neighbour number ,Distance

1

1

2

2

4

2
```

```
enter the no.of neighbours for node ::4

2

enter Neighbour number ,Distance

3

2

5

2

enter the no.of neighbours for node ::5

2

enter Neighbour number ,Distance

1

3

4

2



    matrix


     0     2     1     99     3


     2     0     2     99     99


     1     2     0     2      99


     99     99     2     0     2
```

```
        3      99      99      2       0


to find the path

enter starting node

1

enter ending node

4

Shortest Path is:

1->3->4->

Minimum cost is 3
```

Output2:-

```
enter no.of nodes

4

enter the no.of neighbours for node ::1

2

enter Neighbour number ,Distance

3

2

4

2

enter the no.of neighbours for node ::2

4

enter Neighbour number ,Distance

1
```

```
3

3

1

1

2

4

1

enter the no.of neighbours for node ::3

1

enter Neighbour number ,Distance

4

1

enter the no.of neighbours for node ::4

4

enter Neighbour number ,Distance

3

1

1

2

2

1

4

0

matrix

0       99      2       2
```

```
2       0       1       1

99      99      0       1

2       1       1       0



to find the path

enter starting node

1

enter ending node

2

Shortest Path is:

1->4->2->

 Minimum cost is 3
```

PROGRAM 8

AIM:*WRITE A JAVA PROGRAM TO IMPLEMENT TAKE AN EXAMPLE SUBNET GRAPH WITH WEIGHTS INDICATING DELAY BETWEEN NODES. NOW OBTAIN ROUTING TABLE AT EACH NODE USING DISTANCE VECTOR ROUTING ALGORITHM.*

DESCRIPTION: Distance Vector means that Routers are advertised as vector of distance and Direction. Direction is simply next hop address and exit interface and Distance means hop count.
Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination.

In distance vector routing, the least cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distance to every node. As the name suggests the DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereasIGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as ⍰routing by rumor⍰ because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

SOURCE CODE: (C Language)

```
#include<stdio.h>

int dist[50][50],temp[50][50],n,i,j,k,x;

void dvr();

int main()

{

   printf("\nEnter the number of nodes : ");

   scanf("%d",&n);

   printf("\nEnter the distance matrix :\n");

   for(i=0;i<n;i++)

   {
```

```
            for(j=0;j<n;j++)

            {

                scanf("%d",&dist[i][j]);

                dist[i][i]=0;

                temp[i][j]=j;

            }

            printf("\n");

            }

      dvr();

      printf("To enter new cost between the nodes enter value of i &j:");

      scanf("%d",&i);

             scanf("%d",&j);

             printf("enter the new cost");

             scanf("%d",&x);

             dist[i][j]=x;

             printf("After update\n\n");

      dvr();

             return 0;

}

void dvr()

{

            for (i = 0; i < n; i++)

                for (j = 0; j < n; j++)

                        for (k = 0; k < n; k++)

                                    if (dist[i][k] + dist[k][j] < dist[i][j])

                                    {

                                    dist[i][j] = dist[i][k] + dist[k][j];
                        temp[i][j] = k;
                        }


        for(i=0;i<n;i++)
        {
          printf("\n\nRouting Table for Router:: %d is \n",i+1);
          printf("To\tCost\tNext\n");
          for(j=0;j<n;j++)
          {
          if ((temp[i][j]+1)==(j+1))
          printf("\n%d\t %d\t --",j+1,dist[i][j]);
          else
              printf("\n%d\t %d\t %d",j+1,dist[i][j],temp[i][j]+1);
              }
        }
   printf("\n\n");


}
```

Output 1:-

```
enter the number of nodes :

5

enter the cost matrix :

0  5  2  3  99

5  0  4  99  3

2  4  0 99  4

3  99  99  0  99
```

```
 99  3  4  99  0

Routing Table for router 1 is

To      Cost    Next

1         0       --

2         5        --

3         2        --

4         3        --

5         6        3

Routing Table for router 2 is

To      Cost    Next

1         5       --

2         0       --

3         4       --

4         8       1

5         3       --

Routing Table for router 3 is

To      Cost    Next

1           2      --

2           4      --

3           0      --

4           5      1

5           4      --

Routing Table for router 4is

To      Cost    Next

1          3      --
```

```
2           8       1

3           5       1

4          0      --

5          9      1

Routing Table for router 5is

To       Cost    Next

1            6       3

2            3       --

3            4       --

4            9       3

5            0       --
```

<u>Output 2:-</u>

```
enter the number of nodes :

4

enter the cost matrix :

2  3  99  99

5  2  7 8

99  4  2  1

6  99  3  6


Routing Table for router 1is

To       Cost    Next

1            0       --

2            3       --
```

```
3           10      2

4           11      2

Routing Table for router 2is

To      Cost    Next

1           5       --

2           0       --

3           7       --

4           8       --

Routing Table for router 3is

To      Cost    Next

1           7        4

2           4        --

3           0        --

4           1        --

Routing Table for router 4is

To      Cost    Next

1         6       --

2         7       3

3         3       --

4         0       --
```

PROGRAM 9

AIM:*WRITE A C PROGRAM TO TAKE AN EXAMPLE AS SUBNET OF HOST TO IMPLEMENT BROADCAST TREE FOR IT*

DESCRIPTION:
Kruskal's algorithm is used to obtain th ε broadcast tree from the given subnet.

This algorithm constructs a minimal spanning tree for a connected weighted graph G.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define VERTEX 5
#define FALSE -1
#define TRUE 1
void create_graph(int [][VERTEX],int [][VERTEX],int []);
void prims(int [][VERTEX],int [][VERTEX],int []);
void display(int [][VERTEX]);
void main(){
int graph[VERTEX][VERTEX];
int tree[VERTEX][VERTEX],selected[VERTEX];
clrscr();
create_graph(graph,tree,selected);
prims(graph,tree,selected);
display(tree);
}
void create_graph(int graph[][VERTEX],int tree[][VERTEX],int selected[]){
int i,j;
printf("\nIf there is No Edge B/W Two Nodes,It's Weight is 99");
for(i = 0; i < VERTEX; i++) {
for(j = 0; j < VERTEX; j++) {
printf("\nEnter weight between %d Node %d Node",i,j);
scanf("%d",&graph[i][j]);
tree[i][j] = 0;
}}
for(i = 0; i<VERTEX; i++) {
selected[i] = FALSE;
}}
```

```
void prims(int graph[][VERTEX], int tree[][VERTEX], int selected[VERTEX]){
int ne = 1;
int x,y;
selected[0] = TRUE;
while(ne < VERTEX) {
int min = 99,i,j;
for(i = 0; i < VERTEX; i++){
if(selected[i] == TRUE) {
for(j = 0; j<VERTEX; j++) {
if(selected[j] == FALSE) {
if(min > graph[i][j]){
min = graph[i][j];
x = i;
y = j;
}
}
}
} }
tree[x][y] = 1;
tree[y][x] = 1;
selected[y] = TRUE;
ne++;
}
}
void display(int tree[][VERTEX])
{
int i,j;
printf("%6c",' ');
for(i =0;i<VERTEX;i++)
printf("%6d",i+1);
printf("\n");
for( i = 0; i<VERTEX; i++)
{
printf("%6d",i+1) ;
for(j = 0; j<VERTEX; j++)
printf("%6d",tree[i][j]);
printf("\n");
}
}
```

Output :-

```
If there is No Edge B/W Two Nodes,It's Weight is 99
Enter weight between 0 Node 0 Node
Enter weight between 1 Node 0 Node5
Enter weight between 1 Node 1 Node99
Enter weight between 1 Node 2 Node4
Enter weight between 1 Node 3 Node99
Enter weight between 1 Node 4 Node3
Enter weight between 2 Node 0 Node2
Enter weight between 2 Node 1 Node4
Enter weight between 2 Node 2 Node99
Enter weight between 2 Node 3 Node4
Enter weight between 2 Node 4 Node4
Enter weight between 3 Node 0 Node3
Enter weight between 3 Node 1 Node99
Enter weight between 3 Node 2 Node99
Enter weight between 3 Node 3 Node99
Enter weight between 3 Node 4 Node99
Enter weight between 4 Node 0 Node99
Enter weight between 4 Node 1 Node3
Enter weight between 4 Node 2 Node4
Enter weight between 4 Node 3 Node99
Enter weight between 4 Node 4 Node99
       1    2    3    4    5
   1   0    0    1    1    0
   2   0    0    1    0    1
   3   1    1    0    0    0
   4   1    0    0    0    0
   5   0    1    0    0    0
```

PROGRAM 10

AIM:***WRITE A C PROGRAM TO USING RSA ALGORITHM ENCRYPT A TEXT DATA AND DECRYPT THE SAME:***

DESCRIPTION:
RSA is an Internet encryption and authentication system that uses analgorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The RSA algorithm is the most commonly used encryption and authentication algorithm and is included as part of the Web browsers from Microsoft and Netscape.

SOURCE CODE:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
        int m,n,d,e,val,count1,count2,p,q,i,j,k,pn,a[10],op;
        int c=1;        clrscr();
        printf("\nRSA ALGORITHM");
        printf("\nEnter the Plain Text :");
        scanf("%d",&m);
        printf("\nKey Generation\n");
        for(p=1;p<=180;p++)
        {
        for(q=1;q<=180;q++)
        {
                if(p==q)
                goto one;
                if(p*q>m)
                {
                        count1=0;
                        for(i=1;i<=p;i++)
                        if(p%i==0)
                        count1++;
                        count2=0;
                        for(i=1;i<=q;i++)
                        if(q%i==0)
                        count2++;
                        if((count1==2)&&(count2==2))
                        goto two;
                }
one:
        }
        }
two:
        printf("\n1.prime numbers p=%d\tq=%d",p,q);
        n=p*q;
        printf("\n2.n=%d",n);
        pn=(p-1)*(q-1);
        printf("\n3.pn=%d",pn);
        for(e=1;e<=180;e++)
        {
                count1=0;
                for(i=1;i<=e;i++)
                        if(e%i==0)
                        count1++;
                if(count1==2)
                val=gcd(pn,e);
                if(val==1)
                goto three;
        }
three:

        printf("\n4.e=%d",e);
        for(d=1;d<=180;d++)
        if((d*e)%pn==1)
        goto four;
four:   printf("\n5.d=%d",d);
        printf("\nEncryption \nCipher text is :");
        k=1;
        for(i=1;i<=(e/2);i++)
        {
        c=((int)pow(m,2))%n;
        k=k*c;
        k=k%n;
        }
        if(e%2!=0)
        k=k*(m%n);
        c=k%n;
        printf("%d",c);
        printf("\nDecryption\nPlain text is :");
        k=1;

        for(i=1;i<=(d/2);i++)
        {
        op=((int)pow(c,2))%n;
        k=k*op;
        k=k%n;
        }
        if(d%2!=0)
        k=k*(c%n);
        printf("%d",k%n);
        getch();
}

int gcd(int pn,int e)
{
        int c,j;
        for(j=1;j<=pn;j++)
        {
                if((pn%j==0)&&(e%j==0))
                c=j;
        }
        return c;
}
```

Output:-

```
RSA ALGORITHM
Enter the Plain Text :56
Key Generation
1.prime numbers p=2     q=29
2.n=58
3.pn=28
4.e=3
5.d=19
Encryption
Cipher text is :50
Decryption
Plain text is :56
```

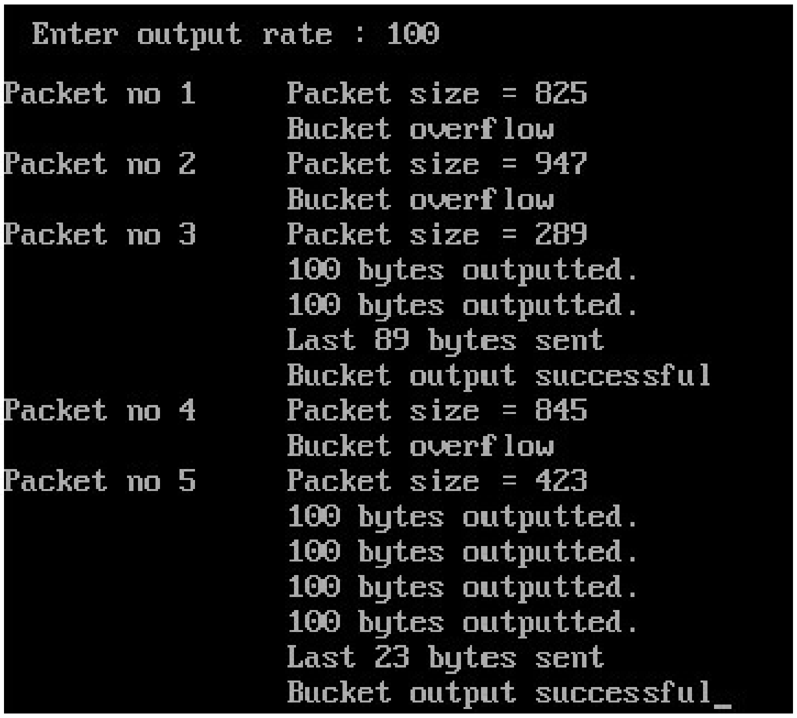
PROGRAM 11 :

AIM:***WRITE A PROGRAM FOR CONGESTION CONTROL USING LEAKY BUCKET ALGORITHM.***

DESCRIPTION:

SOURCE CODE:

```
#include<iostream.h>
#include<dos.h>
#include<stdlib.h>
void bktInput(int,int)
#define bucketSize 512
void main()
 {
int oprate, pktSize;
randomize();
cout<<"Enter output rate : ";
cin>>oprate;
for(int i=1;i<=5;i++)
{
delay(random(1000));
pktSize=random(1000);
cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
bktInput(pktSize,oprate);
}
}
        void bktInput(int ps,int or)
         {
        if(ps>bucketSize)
        cout<<"\n\t\tBucket overflow";
        else
        {
        delay(500);
        while(ps>or)
        {
        cout<<"\n\t\t"<<or<<" bytes outputted.";
        ps=ps-or;
        delay(500);
        }
        if (ps>0)
        cout<<"\n\t\tLast "<<ps<<" bytes sent\t";
        cout<<"\n\t\tBucket output successful";
        }
}
```



```
 Enter output rate : 100

Packet no 1      Packet size = 825
                 Bucket overflow
Packet no 2      Packet size = 947
                 Bucket overflow
Packet no 3      Packet size = 289
                 100 bytes outputted.
                 100 bytes outputted.
                 Last 89 bytes sent
                 Bucket output successful
Packet no 4      Packet size = 845
                 Bucket overflow
Packet no 5      Packet size = 423
                 100 bytes outputted.
                 100 bytes outputted.
                 100 bytes outputted.
                 100 bytes outputted.
                 Last 23 bytes sent
                 Bucket output successful_
```

PROGRAM 12

AIM: ***WRITE PROGRAM FOR FRAME SORTING TECHNIQUE USED IN BUFFERS.***

SOURCE CODE:

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define FSize 3
typedef struct packet{int SeqNum; char Data[FSize+1];}packet;
struct packet *readdata, *transdata;
int divide(char *msg) {
int msglen, NoOfPacket, i, j;
msglen = strlen(msg);
NoOfPacket = msglen/FSize;
if ((msglen%FSize)!=0) NoOfPacket++;
readdata = (struct packet *)malloc(sizeof(packet) * NoOfPacket);
for(i = 0; i < NoOfPacket; i++) {
readdata[i].SeqNum = i + 1;
for (j = 0; (j < FSize) && (*msg != '\0'); j++, msg++)
readdata[i].Data[j] = *msg;
readdata[i].Data[j] = '\0';
}
printf("\nThe Message has been divided as follows\n");
printf("\nPacket No. Data\n\n");
for (i = 0; i < NoOfPacket; i++)
printf(" %2d %s\n", readdata[i].SeqNum,
readdata[i].Data);
return NoOfPacket;
}
void shuffle(int NoOfPacket) {
int *Status;
int i, j, trans;
randomize();
Status=(int * )calloc(NoOfPacket, sizeof(int));
transdata = (struct packet *)malloc(sizeof(packet) * NoOfPacket);
for (i = 0; i < NoOfPacket;) {
```

```
trans = rand()%NoOfPacket;
```

```
if (Status[trans]!=1) {
transdata[i].SeqNum = readdata[trans].SeqNum;
strcpy(transdata[i].Data, readdata[trans].Data);
i++; Status[trans] = 1;
}
}
free(Status);
}
void sortframes(int NoOfPacket) {
packet temp;
int i, j;
for (i = 0; i < NoOfPacket; i++)
for (j = 0; j < NoOfPacket – i-1; j++)
if (transdata[j].SeqNum > transdata[j + 1].SeqNum) {
temp.SeqNum = transdata[j].SeqNum;
strcpy(temp.Data, transdata[j].Data);
transdata[j].SeqNum = transdata[j + 1].SeqNum;
strcpy(transdata[j].Data, transdata[j + 1].Data);
transdata[j + 1].SeqNum = temp.SeqNum;
strcpy(transdata[j + 1].Data, temp.Data);
}
}
void receive(int NoOfPacket) {
int i;
printf("\nPackets received in the following order\n");
for (i = 0; i <NoOfPacket; i++) printf("%4d", transdata[i].SeqNum);
sortframes(NoOfPacket);
printf("\n\nPackets in order after sorting..\n");
for (i = 0; i < NoOfPacket; i++) printf("%4d", transdata[i].SeqNum);
printf("\n\nMessage received is :\n");
for (i = 0; i < NoOfPacket; i++) printf("%s", transdata[i].Data);
}
void main() {
char *msg;
int NoOfPacket;
clrscr();
printf("\nEnter The message to be Transmitted :\n");
```

```
scanf("%[^\n]", msg);
NoOfPacket = divide(msg);
shuffle(NoOfPacket);
receive(NoOfPacket);
free(readdata);
free(transdata);
getch();
}
```

Output

```
Enter The messgae to be Transmitted :
hi, it was nice meeting u on Sunday

The Message has been divided as follows

Packet No.   Data
1            hi,
2            it
3            wa
4            s n
5             ice
6            me
7            eti
8            ng
9            u o
10           n s
11           und
12            ay

Packets received in the following order

4  2  6  3  5  1  8  9  11  7  12  10

Packets in order after sorting..

1  2  3  4  5  6  7  8  9  10  11  12

Message received is :

hi, it was nice meeting u on sunday
```

PROGRAM 13

AIM:WORK WITH WIRESHARK TOOL FOR

i. **PACKET CAPTURE USING WIRE SHARK**
ii. **STARTING WIRE SHARK**
iii. **VIEWING CAPTURED TRAFFIC**
iv. **ANALYSIS AND STATISTICS & FILTERS.**

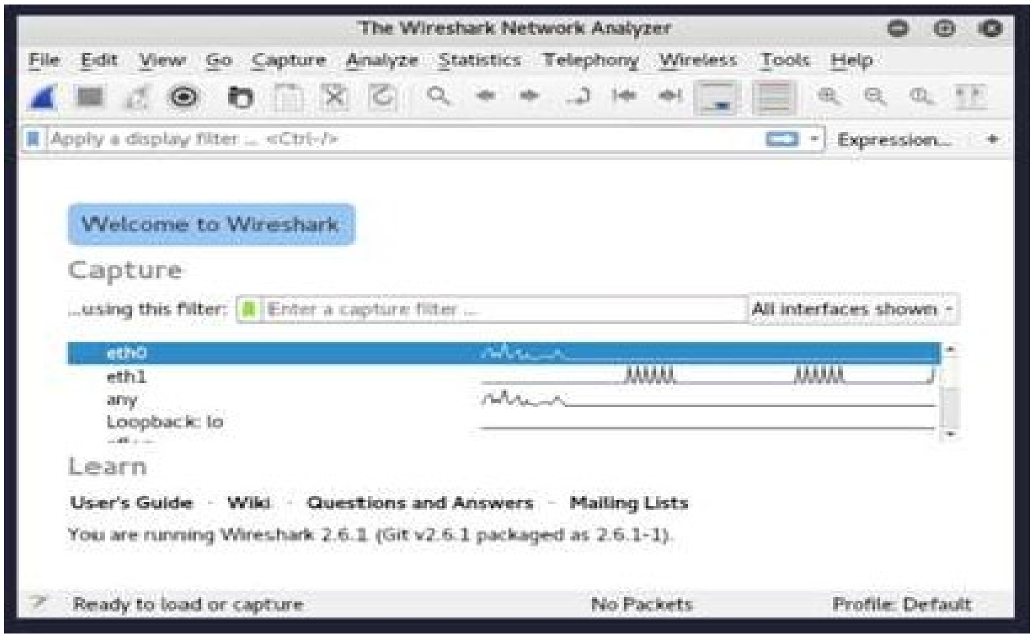
DESCRIPTION:

- Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.
- Wireshark is an open-source packet analyzer, which is u sed for **education, analysis, software development, communication protocol development, and network troubleshooting**.
- It is used to track the pac kets so that each one is filtered to meet our specific needs. It is commonly called as a **sni fer, network protocol analyzer, and network analyzer**. It is also used by network sec urity engineers to examine security pro lems.

Command :

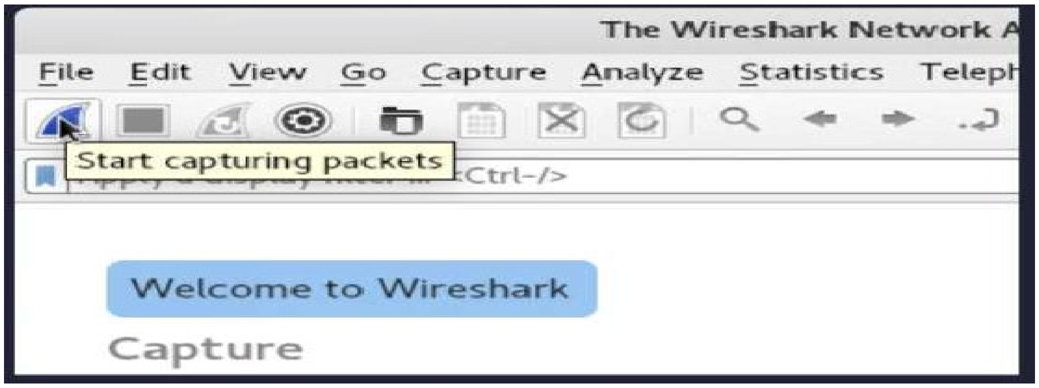
- **Installation in Ubuntu** - `sudo apt-get install wireshark`

➢ **Capturing Data Packets c n Wireshark.**

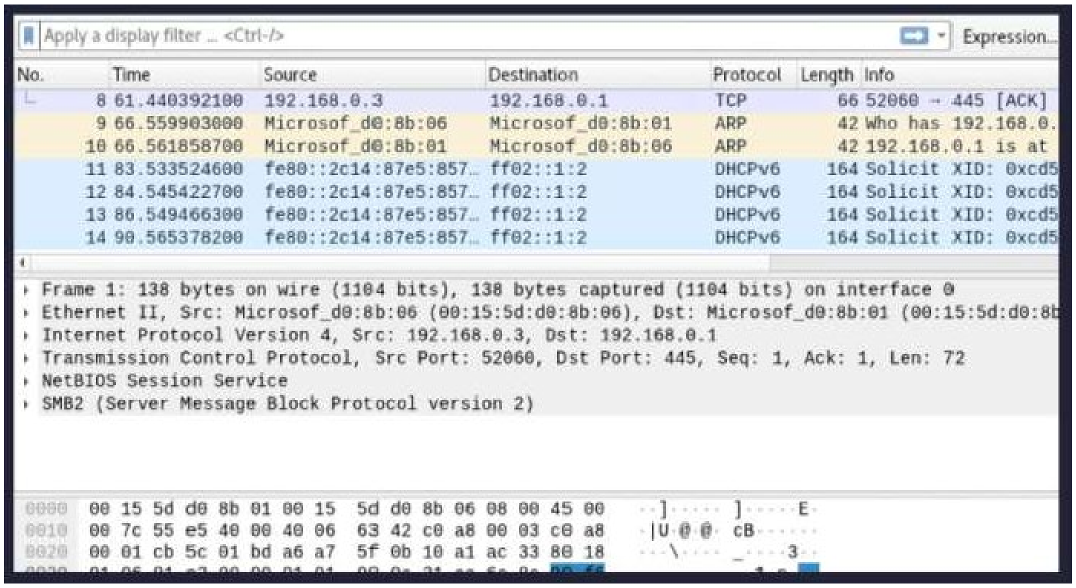

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
eth0
eth1
any
Loopback: lo
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.6.1 (Git v2.6.1 packaged as 2.6.1-1).

Ready to load or capture — No Packets — Profile: Default



➢ **Click the first button on the toolbar, titled "Start Capturing Packets :**



```
No.   Time          Source                   Destination          Protocol  Length  Info
    8 61.440392100  192.168.0.3              192.168.0.1          TCP           66  52060 → 445 [ACK]
    9 66.559903000  Microsof_d0:8b:06        Microsof_d0:8b:01    ARP           42  Who has 192.168.0.
   10 66.561858700  Microsof_d0:8b:01        Microsof_d0:8b:06    ARP           42  192.168.0.1 is at
   11 83.533524600  fe80::2c14:87e5:857…     ff02::1:2            DHCPv6       164  Solicit XID: 0xcd5
   12 84.545422700  fe80::2c14:87e5:857…     ff02::1:2            DHCPv6       164  Solicit XID: 0xcd5
   13 86.549466300  fe80::2c14:87e5:857…     ff02::1:2            DHCPv6       164  Solicit XID: 0xcd5
   14 90.565378200  fe80::2c14:87e5:857…     ff02::1:2            DHCPv6       164  Solicit XID: 0xcd5

› Frame 1: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0
› Ethernet II, Src: Microsof_d0:8b:06 (00:15:5d:d0:8b:06), Dst: Microsof_d0:8b:01 (00:15:5d:d0:8b
› Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.1
› Transmission Control Protocol, Src Port: 52060, Dst Port: 445, Seq: 1, Ack: 1, Len: 72
› NetBIOS Session Service
› SMB2 (Server Message Block Protocol version 2)

0000  00 15 5d d0 8b 01 00 15  5d d0 8b 06 08 00 45 00   ··]·····]·····E·
0010  00 7c 55 e5 40 00 40 06  63 42 c0 a8 00 03 c0 a8   ·|U·@·@·cB······
0020  00 01 cb 5c 01 bd a6 a7  5f 0b 10 a1 ac 33 80 18   ···\····_····3··
```

➢ **Wireshark will show you the packets that it captures in real-time :**

Here are some details about each column in the top pane :

- **No.**: This is the number order of the packet that got captured. The bracket indicates that this packet is part of a c cnversation.
- **Time**: This column shows you how long after you started the packet got captured. You can change this value in the Settings something different displayed.
  - **Source**: This is the address of the system that sent the packet. apture that this menu if you need
- **Destination**: This is the a dress of the destination of that packet.
- **Protocol**: This is the type of packet, for example, TCP, DNS, DHCPv6, or ARP.
- **Length**: This column shows you the length of the packet in bytes.
- **Info**: This column shows you more information about the packet contents, and will vary depending on what kind of packet it is.

PROGRAM 14

AIM:WITH COMMANDS SHOW HOW TO RUN NMAP SCAN AND OPERATING SYSTEM DETECTION USING NMAP

DESCRIPTION:

- Nmap ("Network Mapper") is an open source tool for network exploration and security auditing.
- It was designed to rapidly scan large networks, although it works fine against single posts.
- Nmap uses raw IP packet the network, what service what operating systems (

in novel ways to determine what hosts are available on (application name and version) those hosts are offering, nd OS versions) they are running, what type of packet

filters/firewalls are in use, and dozens of other characteristics.

- While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.
  - **To install NMAP on Ubuntu, run the command :**

```
sudo apt-get install nmap
```

  - **Scan IP range or subnet**

```
sudo nmap target_IP or domain.com
```

  - Instead of scanning individual IPs, scan a whole IP range by defining it in your command line:

Ex : `sudo nmap 185.52.53.2-222`

  - The following command scans the entire specified subnet:

Ex : `sudo nmap 185.52.53.0/24`

- **Port Scanning with Nmap**

  - Nmap is an efficient port scanner that recognizes six port states:
    - open – actively accepting TCP connections, UDP datagrams or SCTP associations
    - closed – accessible; however, no application is listeni g on the port
    - filtered – nma packet filterin cannot determine whether

the port is
open due to

- unfiltered – the port is accessible; however, nmap is determine if it is open or closed unable toopen|filtered – nmap cannot determine if a port is open or filtered
- closed|filtered – nmap cannot establish if a port is cl sed or filtered

➢ **Port Specification and Scan Order**

The **-p** option allows you to specify port ranges and sequences:

```
Ex : sudo nmap -p 80,443 185.52.53.222
```

- **Aggressive Scan**

➢ The **-A** option enables a comprehensive set of scan options. It enables:
➢ OS (Operating System) detection, also available with **O** command
➢ version scanning, also available with the **-sV** command the **-**
➢ script scanning, also available with the **-sC** command
➢ traceroute, also available with the **–traceroute** command

PROGRAM 15

AIM:DO THE FOLLOWING USING NS2 SIMULATOR

i. **NS2 SIMULATOR-INTRODUCTION**
ii. **SIMULATE TO FIND THE NUMBER OF PACKETS DROPPED**
iii. **SIMULATE TO FIND THE NUMBER OF PACKETS DROPPED BY TCP/UDP**
iv. **SIMULATE TO FIND THE NUMBER OF PACKETS DROPPED DUE TO CONGESTION**
v. **SIMULATE TO COMPARE DATA RATE & THROUGHPUT.**
vi. **SIMULATE TO PLOT CONGESTION FOR DIFFERENT SOURCE/DESTINATION**
vii. **SIMULATE TO DETERMINE THE PERFORMANCE WITH RESPECT TO TRANSMISSION OF PACKETS.**

DESCRIPTION:

Network Simulator version 2 (NS-2) is discrete event packet level simulator. The network simulator covers a very large number of application of different kind of protocols of different network types consisting of different network elements and traffic models. NS-2 is a package of tools that simulates behavior of networks such as creating network topologies, log events that happen under any load, analyze the events and understand the network.
The aim of this first experiment is to learn how to use NS-2, to get acquainted with the simulated objects and understand the operations of network simulation. We will also look at how to analyze the outcome of a simulation

Creating a tcl(Tool Command Language) file:

1. Create a simulation process
2. Create nodes
3. Create links
4. Create events(transfer of data….)

TOPOLOGY: Define a topology

1. With four nodes.
2. One node acts as router that forwards the data.
3. Two nodes have to send data to the fourth node.
4. Distinguish the data flows from the two nodes from each other.
5. Show how a queue can be monitored to see how full it is, and how many packets are being discarded.

SOURCE CODE:

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
          #Close the trace file
        close $nf
          #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0


$ns connect $udp0 $null0
$ns connect $udp1 $null0

$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
```

```
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"

$udp0 set class_ 1
$udp1 set class_ 2

$ns color 1 Blue
$ns color 2 Red

$ns duplex-link-op $n2 $n3 queuePos 0.5
```

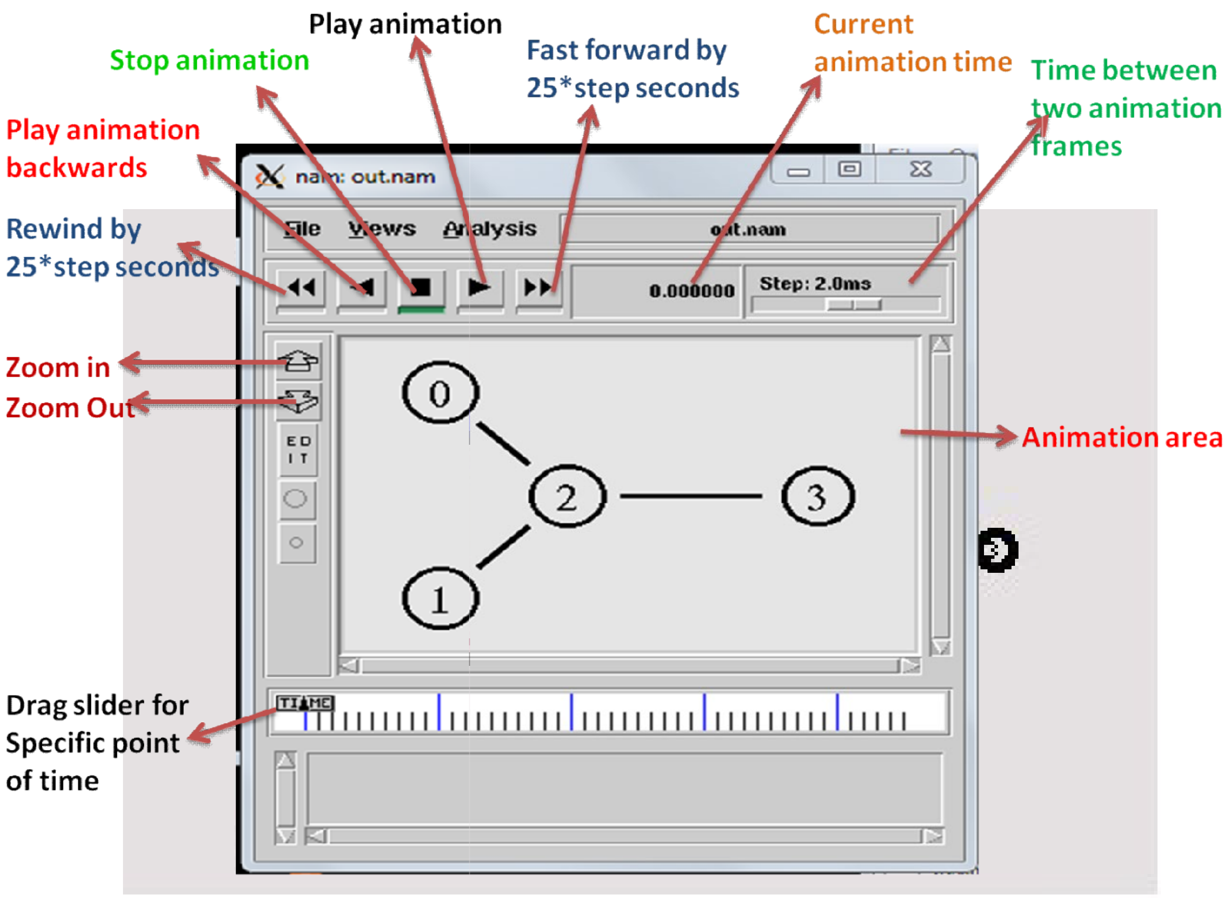
# Insert your own code for topology creation and agent definitions, etc. here

#Call the finish procedure after 5 seconds simulation time

```
$ns at 5.0 "finish"
```

#Run the simulation

```
$ns run
```



PROGRAM 16

AIM:***WRITE A C PROGRAM* implement Hierarchial Routing**

DESCRIPTION:
Routing that is based on hierarchical addressing.
Note: Most Transmission Control Protocol/Internet Protocol (TCP/IP) routing is based on a two-level hierarchical routing in which an IP address is divided into a network portion and a host portion. Gateways use only the network portion until an IP datagram reaches a gateway that can deliver it directly. Additional levels of hierarchical routing are introduced by the addition of subnetworks.

SOURCE CODE:

```
# include<stdio.h>

#include<conio.h>

#include<string.h>

#include<ctype.h>

struct hierar

{

char node[10];

char line[10];

int hops;

}

state[20];

void main()

{

int i,boo,small,t1,t2,j,n;

char ch,rnode[10],temp[10];

clrscr();

printf("Enter number of nodes");

scanf("%d",&n);

printf("Enter %d node,line,hops",n);
```

```
for(i=0;i<n;i++)

scanf("%s %s %d",&state[i].node,&state[i].line,&state[i].hops);

printf(" \n node \t \t line \t \t hops");

for(i=0;i<n;i++)

printf("%s \t \t %s \t \t %d \n",state[i].node,state[i].line,state[i].hops);

printf("Enter starting point");

scanf("%s",&rnode);

printf("\n node \t \t line \t \t hops \n");

for(i=0;i<n;i++)

{

t1=toascii(rnode[0]);

t2=toascii(state[i].node[0]);

if(t1=t2)

boo=0;

else

boo=1;

if(boo=0)

printf("%s \t \t %s \t \t %d \n",state[i].node,state[i].line,state[i].hops);

else

{

printf("%c",state[i].node[0]);

printf("\t \t %s",state[i].line);

strcpy(temp,state[i].node);

small=state[i].hops;

for(j=i;temp[0]==state[i].node[0];j++)
{
if(state[j].hops<small)
small=state[j].hops;
i++;
}
printf("\t \t %d \n",small);
}
}
getch();
}
```

INPUT AND OUTPUT:

```
Enter number of nodes 7
Enter 7 node,line,hops
1A
1B    1B    1
1C    1C    1
2A    1B    2
2B    1B    3
2C    1B    3
2D    1B    4
NODE        LINE        HOPS
1A           -           0
1B          1B           1
1C          1C           1
2A          1B           2
```

```
2B          1B           3

2C          1B           3

2D          1B           4


Enter starting point 1A


NODE            LINE        HOPS

1A              -           0

1B              1B          1

1C              1C          1

2               1B          2
```