## Data Communications:

Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable.

The effectiveness of a data communications system depends on four fundamental characteristics:

1. **Delivery** The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.
2. **Accuracy** The system must deliver the data accurately. Data should not be altered. If the data is altered in transmission and left uncorrected are unusable.
3. **Timeliness** The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced and without significant delay. This kind of delivery is called *real-time* transmission.
4. **Jitter** It refers to the variation in the packet arrival time. Jitter is the uneven delay in the delivery of audio or video packets.

   Example: Let us assume that video packets are sent every 3ms. If some of the packets arrive with 3ms delay and others with 4ms delay, an uneven quality in the video is the result.

## Components:

A data communications system has five components:

1. Message: The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. Sender: The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
3. Receiver: The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. Transmission medium: The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
5. Protocol: A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a

person speaking French cannot be understood by a person who speaks only Japanese.

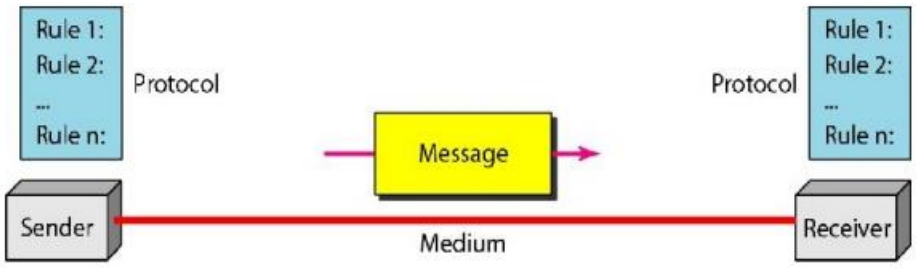



Fig 1: Components of Data Communication.

**Direction of Data flow:**

Communication between two devices can be simplex, half-duplex, or full-duplex.

**Simplex**: The communication is unidirectional, as on a one-way street.

Only one of the two devices on a link can transmit; the other can only receive.

Keyboards and traditional monitors are examples of simplex devices.

The simplex mode can use the entire capacity of the channel to send data in one direction as shown in **Fig 2.a**

**Half-Duplex**: Each station can both transmit and receive, but not at the same time.

The half-duplex mode is like a one-lane road with traffic allowed in both directions.

The entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time as shown in **Fig 2.b**

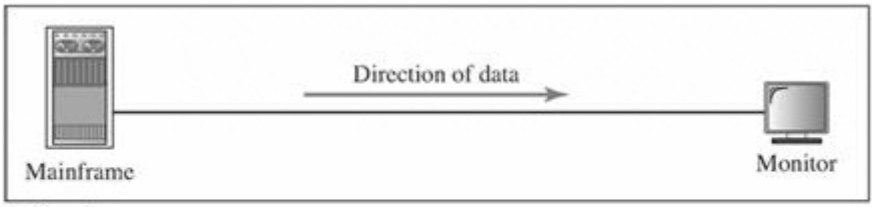
Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

**Full-Duplex**:

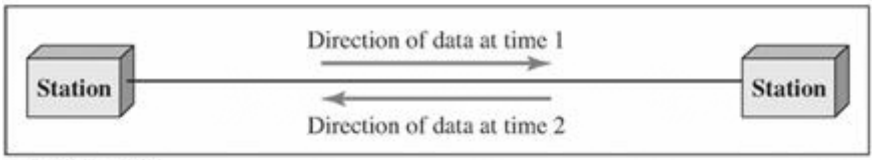
Both stations can transmit and receive simultaneously.

Signals going in one direction share the capacity of the link with signals going in the other direction as shown in **Fig 2.c.**

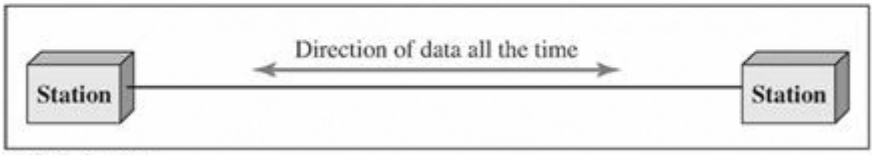
One common example of full-duplex communication is the telephone network.




**a. Simplex**




**b. Half-duplex**




**c. Full-duplex**

## NETWORKS

A **Network** is a set of devices (also called as nodes) connected by communication links. (or)
A **Network** is two or more devices connected through links.

A **Node** can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

A **Link** is a communications pathway that transfers data from one device to another.

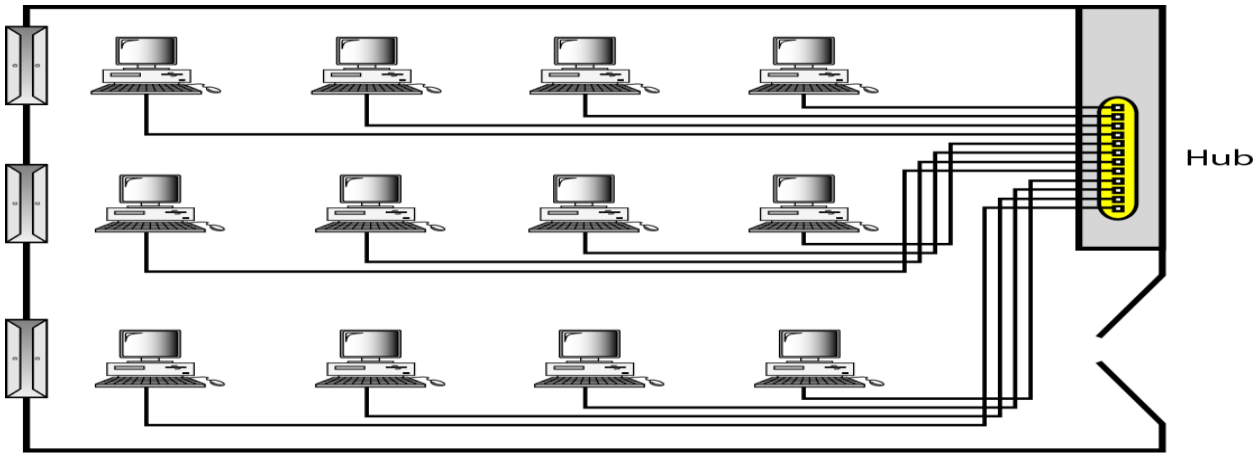
## CATEGORIES OF NETWORKS

There are 3 categories of networks depend on its size:

1. Local Area Networks(LAN)
2. Metropolitan Area Networks(MAN)
3. Wide Area Networks(WAN)

## Local Area Networks

- A Local Area Network (LAN) provides short-distance transmission of data over small geographic areas that may comprise a single office, building, or campus.
- **Size:** LAN size is limited to a few kilometers.
- **Speed:** Early LANs had data rates in the 4 to 16 megabits per second (Mbps) range but now speeds are increased to 100 or 1000Mbps.
- LANs are designed to allow resources to be shared between personal computers or workstations.
- The resources to be shared can include hardware (e.g., a printer), software (e.g., an application program), or data.
- A local area network (LAN) is usually privately owned.
- LAN will use only one type of transmission medium.
- The most common LAN topologies are bus, ring, and star.



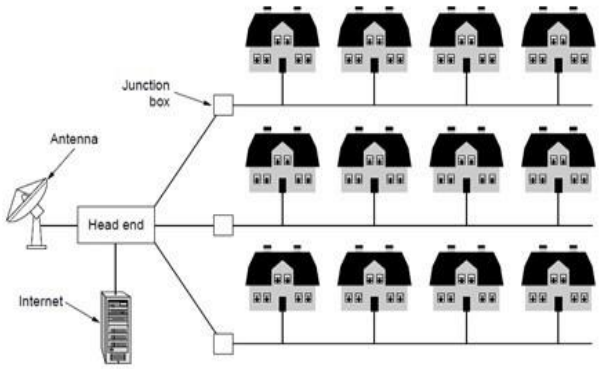

## Metropolitan Area Networks

A Metropolitan Area Network (MAN) is a network with a size between a LAN and a WAN. It normally covers the area inside a town or a city.

It is designed for customers who need a high-speed connectivity to the Internet, and have endpoints spread over a city or part of city.

**Example** of a MAN is the part of the telephone company network that can provide a high- speed DSL line to the customer.

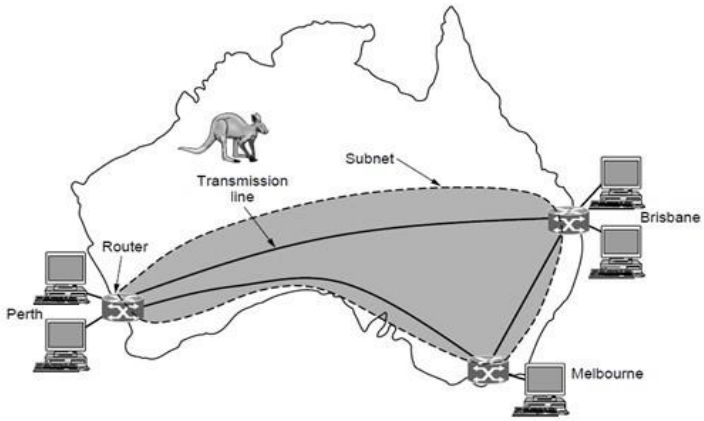
## A metropolitan area network based on cable TV




**Wide Area Network**

A Wide Area Network (WAN) provides long-distance transmission of data, image, audio, and video information over large geographic areas that may comprise a country, a continent, or even the whole world.

## WAN that connects three branch offices in Australia.

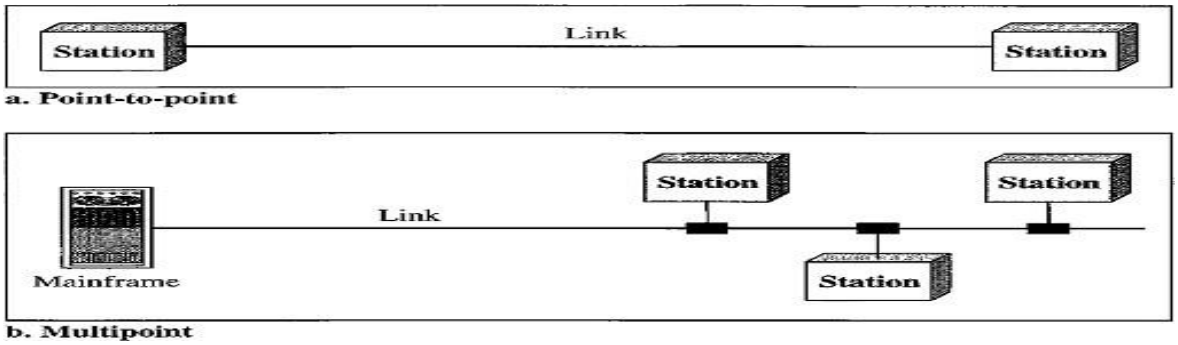



**Type of Connection**

Two devices must be connected in some way to the same link at the same time for occurring of communication. There are two possible types of connections:

1. Point-to-Point Connection

2. Multipoint Connection




**Point-to-Point Connection**

- A Point-to-Point connection provides a dedicated link between two devices.
- The entire capacity of the link is reserved for transmission between those two devices.
- Point-to-Point connections use an actual length of wire or cable to connect the two ends and microwave or satellite links.
- Example: When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

**Multipoint (or) Multi-drop Connection**

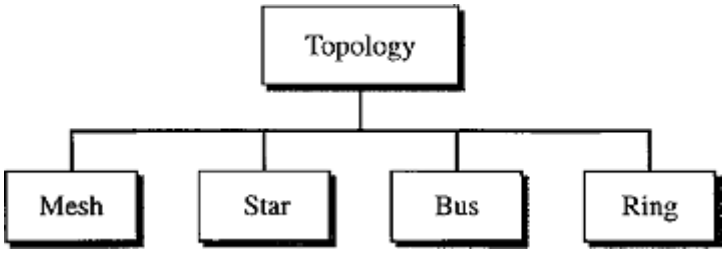
- A multipoint connection is more than two specific devices share a single link.
- In a multipoint environment, the capacity of the channel is shared, either spatially or temporally.
- If several devices can use the link simultaneously, it is a spatially shared connection. If users must take turns, it is a time shared connection.

## NETWORK TOPOLOGIES

The term physical topology refers to the way in which a network is connected physically. Two or more devices connect to a link. Two or more links form atopology.

There are four basic topologies are present:

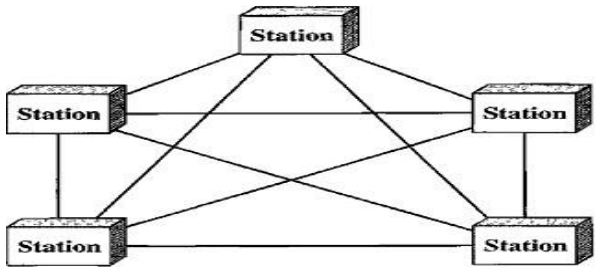
1. Bus
2. Ring
3. Star
4. Mesh




### Mesh Topology

- In a mesh topology, every device has a **Dedicated Point-to-Point** link to every other device. (i.e.) for each node there is a link to all other nodes.
- The term **Dedicated** means that the link carries traffic only between the two devices it connects.

**Advantages:**

1. A mesh topology is robust. If one link becomes unusable, it does not affect the entire system.
2. The use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices.
3. **Privacy or Security.** When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages.
4. Point-to-Point links make **Fault Identification** and **Fault Isolation** easy.
5. 


**Disadvantages:**

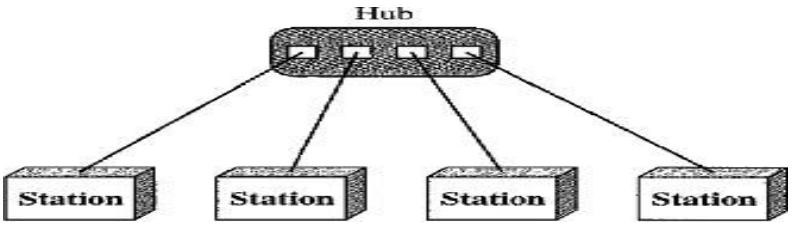
1. **High Cost:** Every device must be connected to every other device then there is a high amount of cabling and huge number of I/O ports required, this will make installation and reconnection are difficult.
2. The hardware required to connect each link (I/O ports and cable) can be prohibitively expensive.
3. More hardware (i.e. cables) and space is required

**Example**: Telephone offices and Police stations.

Connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

**Star Topology**

- In a star topology, each device has a dedicated point-to-point link only to a central controller called a Hub or Switch. The devices are not directly linked to one another.
- A star topology does not allow direct traffic between devices. The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, and the controller transfers the data to the other connected device.




**Advantages:**

1. A star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect it to any number of others. This factor also makes it easy to install and reconfigure
2. Less cabling is required than mesh topology.
3. Star topology is robust, If one link fails, only that link is affected. All other links remain active.

**Disadvantages:**

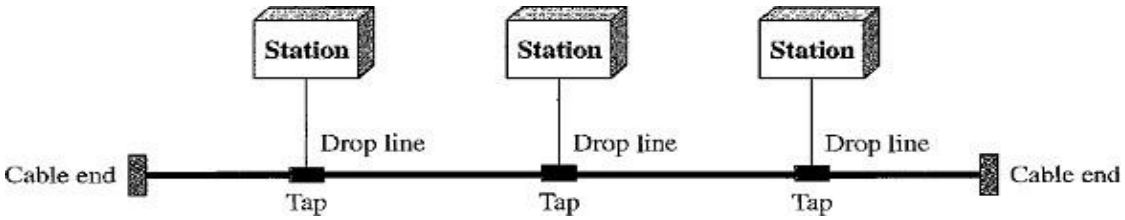
1. If hub fails entire processing will be stopped working.

Uses:

2. It is used in High-speed LAN's often use a star topology with a central hub.

**Bus Topology**

- A **bus topology** is multipoint connection, one long cable acts as a **backbone** to link all the devices in a network. Here the cable is called the bus.
- Bus topology was the one of the first topologies used in the design of early local area networks.
- Nodes are connected to the bus cable by drop lines and taps.
- A drop line is a connection running between the device and the main cable.
- A tap is a connector that splices into (attached to) the main cable.




**Advantages:**

1. Installation is easy. Bus Backbone cable can be laid along the most efficient path and then connected to the nodes by drop lines of various lengths.
2. A bus uses less cabling than mesh or star topologies.

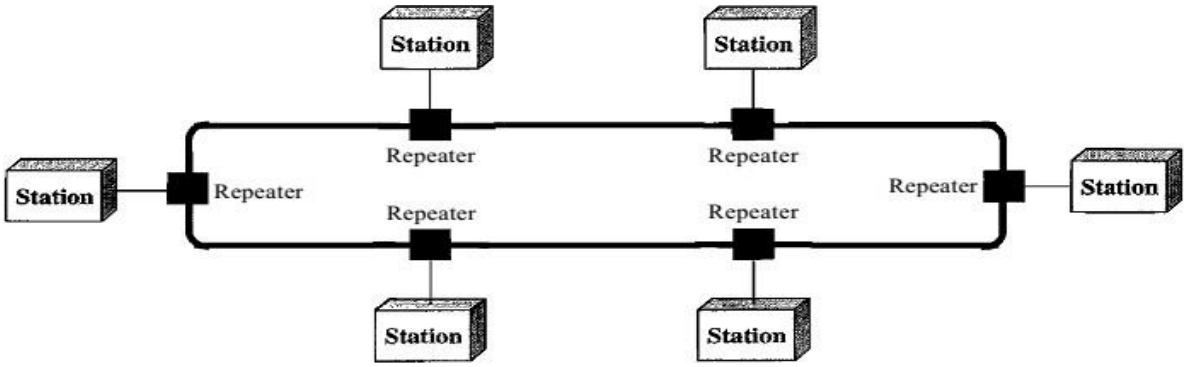
**Disadvantages:**

1. All the devices are connected to bus backbone cable, so that if the backbone cable fails the entire system fails.

2. Difficult Reconnection and Fault Isolation. It is difficult to add new devices.
3. There is a limit on the number of taps a bus can support and on the distance between those taps.
4. More heat is generated if the number of taps is more. Heat degrades the quality of signal.

**Ring Topology**

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




**Advantages:**

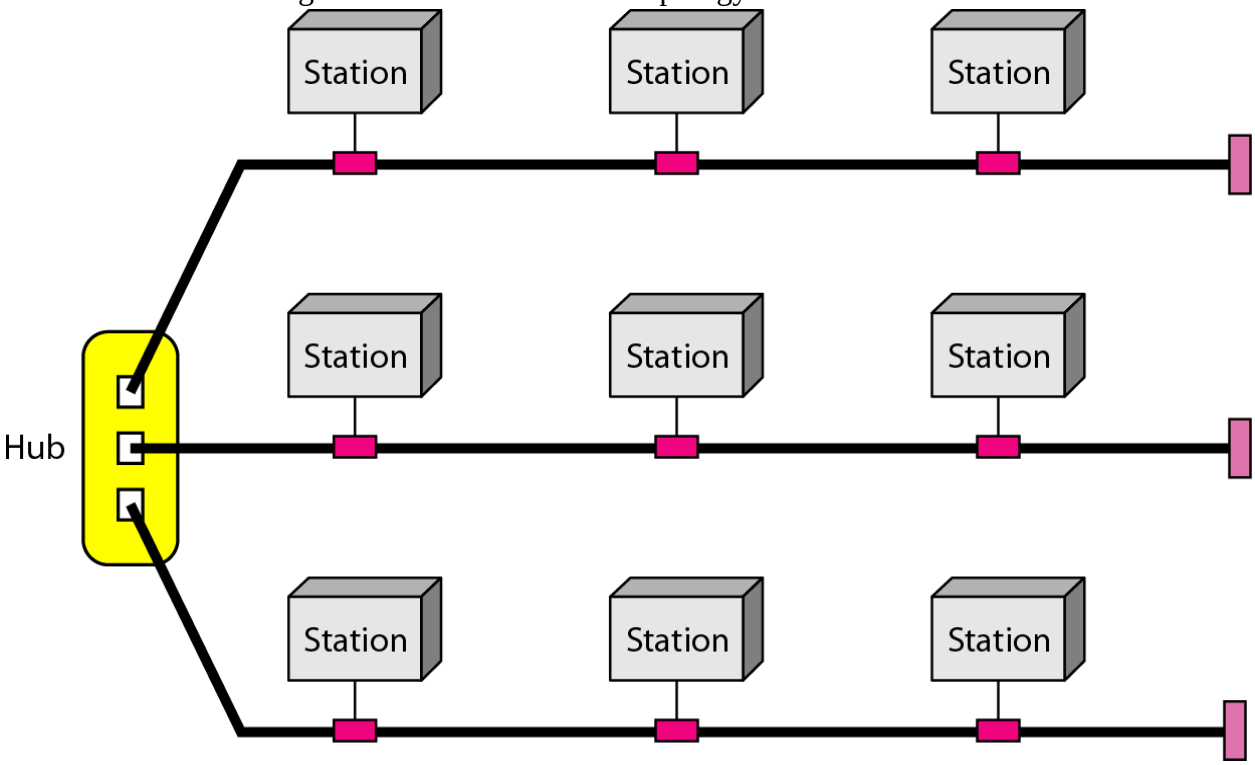
1. A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically).
2. To add or delete a device requires changing only two connections.
3. The only constraints are media and traffic considerations (maximum ring length and number of devices).

**Disadvantage:**

1. Unidirectional traffic can be a disadvantage.
2. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network.

**Hybrid Topology**

It is a combination of two or more topologies for example star topology with each branch connecting several stations in a bus topology



# PROTOCOLS AND STANDARDS

## PROTOCOLS

A protocol is a set of rules that govern data communications. A protocol defines what is communicated, how it is communicated, and when it is communicated. For communication to occur, the entities must agree on a protocol.

The key elements of a protocol are:

1. Syntax
2. Semantics
3. Timing.

**Syntax**

- The term *syntax* refers to the structure or format of the data, meaning the order in which they are presented.
- For example, a simple protocol might expect the first 8 bits of data to be the address of the sender, the second 8 bits to be the address of the receiver, and the rest of the stream to be the message itself.

**Semantics**

- The word *semantics* refers to the meaning of each section of bits. How are a particular pattern to be interpreted, and what action is to be taken based on that interpretation?
- For example, does an address identify the route to be taken or the final destination of the message?

**Timing**

- The term *timing* refers to two characteristics: when data should be sent and how fast they can be sent.
- For example, if a sender produces data at 100 Mbps but the receiver can process data at only 1 Mbps, the transmission will overload the receiver and some data will be lost.

## STANDARDS

Standards provide guidelines to manufacturers, vendors, government agencies, and other service providers to ensure the kind of interconnectivity necessary in today's marketplace and in international communications.

Data communication standards fall into two categories: **de facto** and **de jure**.

### De facto (meaning "by fact" or "byconvention")

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

### De jure (meaning "by law" or "by regulation")

Those standards that have been legislated by an officially recognized body are de jure standards.

**Standards Organizations**

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

Standards Creation Committees

1. ISO (International Organization for Standardization)
2. IEEE (Institute of Electrical and Electronics Engineers)
3. ANSI (American National Standards Institute)
4. ITU-T (International Telecommunication Union-Telecommunication StandardsSector)
5. EIA (Electronic Industries Association)

**ISO/OSI MODEL:**

This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers

The model is called the ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems.

**The OSI model has seven layers.**

The **principles** that were applied to arrive at the seven layers can be summarized as follows:

1. A layer should be created where a different abstraction is needed.

2. Each layer should perform a well-defined function.

3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.

4. The layer boundaries should be chosen to minimize the information flow across the interfaces.

5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

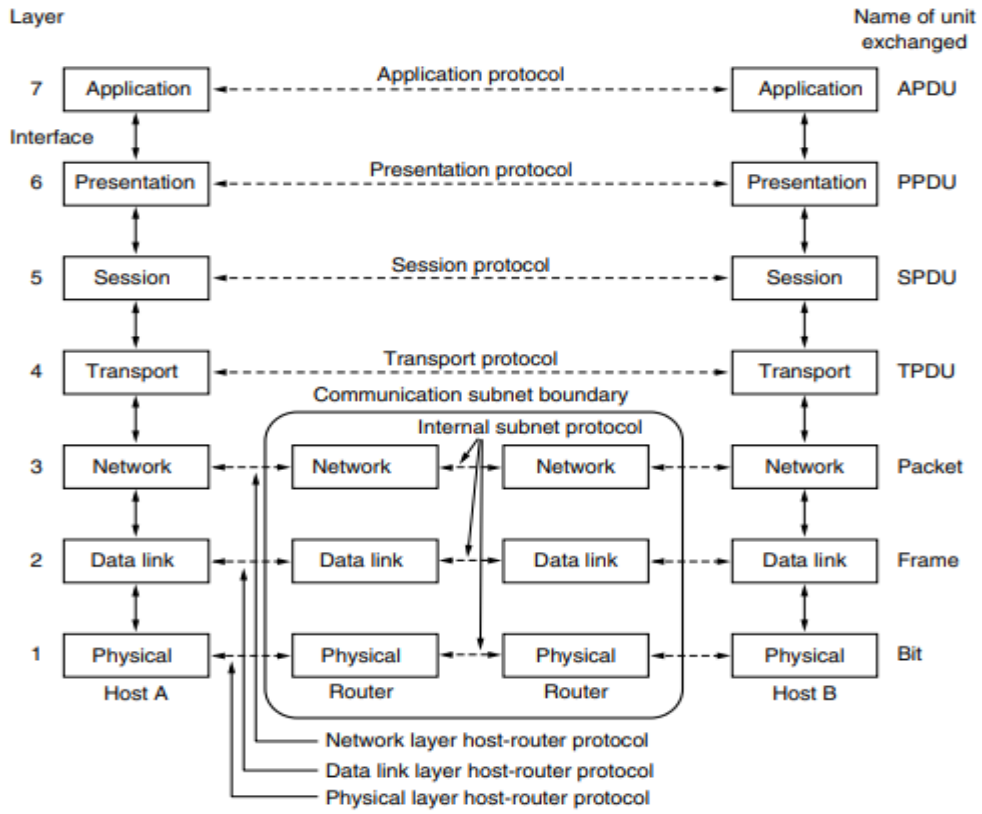



Fig 1: OSI Model

NOTE:

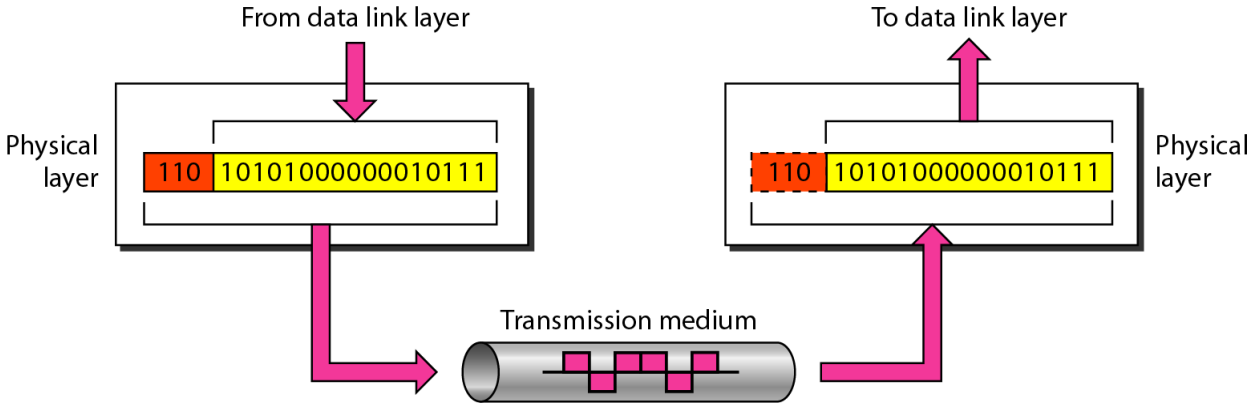
- Layers 1-4 relate to communications technology.
- Layers 5-7 relate to user applications.

## Physical Layer

The **Physical Layer** is concerned with transmitting raw bits over a communication channel.

Physical Layer is responsible for:

- It defines the procedures and functions that physical devices and interfaces have to perform for transmission to occur.
- It also defines the type of transmission medium.
- It defines the data transmission rate, synchronization of data between sender and receiver.
- It defines type of connection (point-to-point or multipoint), type of topology, type of transmission mode, type of dataflow (simplex, half duplex, duplex).




## The Data Link Layer

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

The **main task** of the Data link layer is **Error Free Transmission**. At the sender the data link layer break up the input data into **data frames** and transmits the frames sequentially.

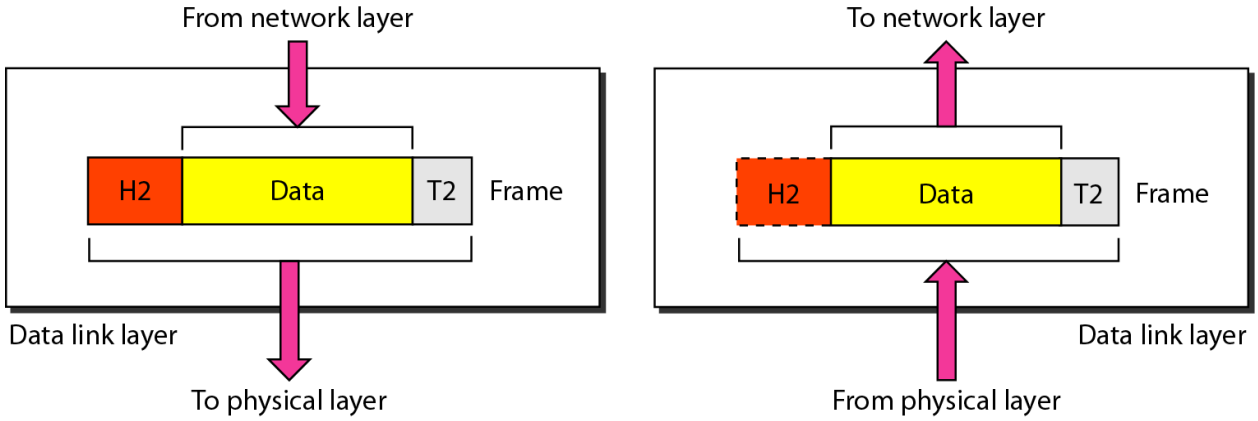
Frame is typically a few hundred or a few thousand bytes.

Other responsibilities of the data link layer include the following:

- **Framing** - The data link layer divides the stream of bits received from the network layer into manageable data units called frames
- **Physical addressing** - If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and receiver of the frame.
- **Flow control** - If the rate at which the data are received by the receiver is less than the rate at which data sent by the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
- **Error control** - The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also

uses a mechanism to recognize duplicate frames. Error control is normally achieved through a trailer added to the end of the frame.

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




## Network Layer

The network layer is responsible for the delivery of individual packets from the source host to the destination host through single or multiple networks.

**Note:** If two systems are connected to the same network then there is usually no need for a network layer.

If the two systems are connected to different networks with connecting devices between the networks then there is a need for the network layer to accomplish source-to-destination delivery.

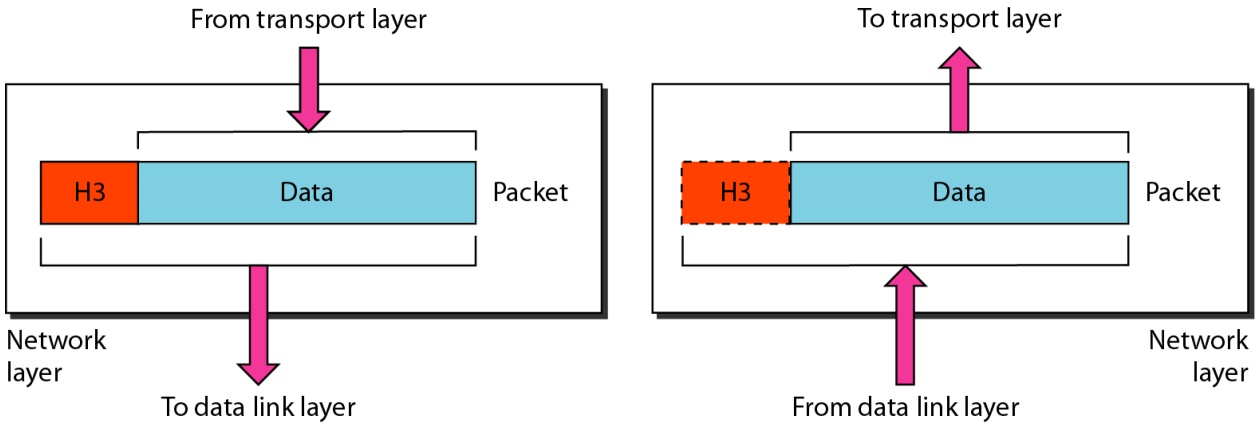
Responsibilities of the Network layer include the following:

**Logical addressing**

- The physical addressing is implemented by Data- link layer, whereas logical addressing is implemented by network layer.
- Data- link layer handles the addressing problem locally, but if packets passes the network boundary there is a need for logical addressing system to help distinguish source and destination systems.
- The network layer adds a header to the packet coming from the upper layer that includes the logical addresses of the sender and receiver.

**Routing**

- When independent networks or links are connected to create inter- networks (network of networks) or a large network, the connecting devices (called *routers* or *switches)* route the packets to their final destination.




## Transport Layer

The transport layer is a true end-to-end layer; it carries from the source to the destination.

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

**Responsibilities of the Transport Layer Include:**

**Port addressing (or) Service point addressing**

- Source-to-Destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other.
- The transport layer header must therefore include a type of address called a *service-point address* (or port address).
- The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on that computer.

**Segmentation and Reassembly**

- A message is divided into transmittable segments, with each segment containing a sequence number.
- These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and the sequence numbers are used for identifying and replace packets that were lost during transmission.

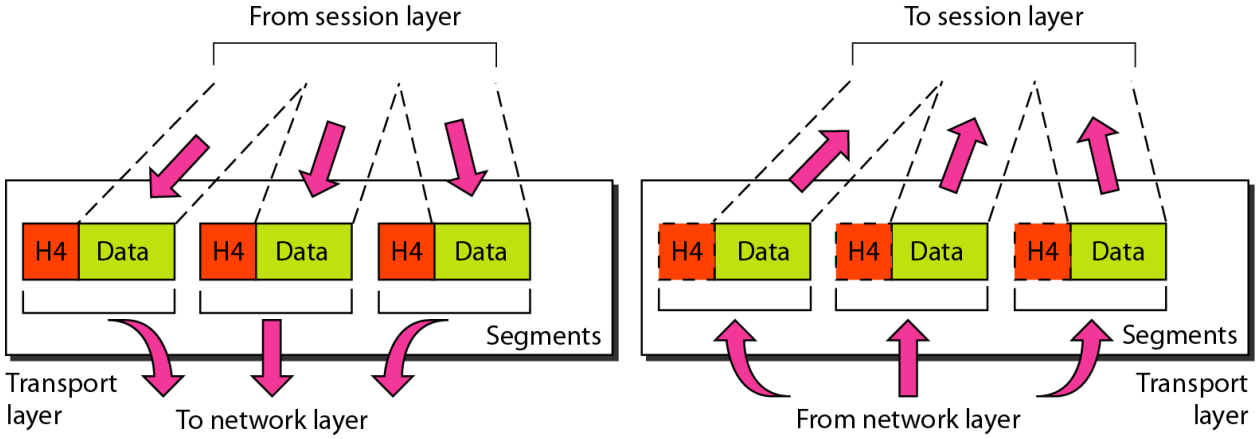
**Connection control**

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

**Flow control and Error control**

- Like the data link layer, the transport layer is responsible for flow control.
- Flow control at this layer is performed end to end rather than across a single link.
- Like the data link layer, the transport layer is responsible for error control.

- Error control at this layer is performed Process-to-Process rather than across a single link.
- Error control achieved through **Retransmission.**

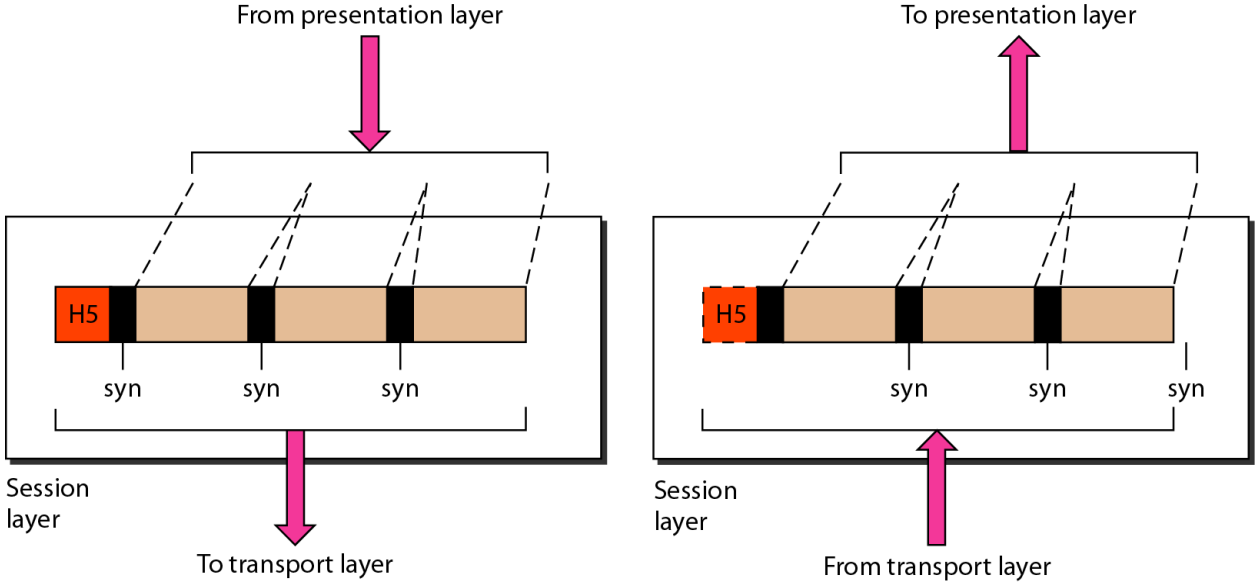



## Session Layer

The session layer allows users on different machines to establish **sessions** between them. The session layer is the network *dialog controller*. It establishes, maintains, and synchronizes the interaction among communicating systems.

**Responsibilities of the session layer include the following**

- **Dialog Control** The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half-duplex (one way at a time) or full-duplex (two ways at a time) mode.
- **Synchronization** The session layer allows a process to add checkpoints, or synchronization points, to a stream of data. Check-Pointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery
- **Token management** prevents two parties from attempting the same critical operation simultaneously.



## Presentation Layer

The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.

The presentation layer is responsible for **Translation, Compression, and Encryption**. **Translation**

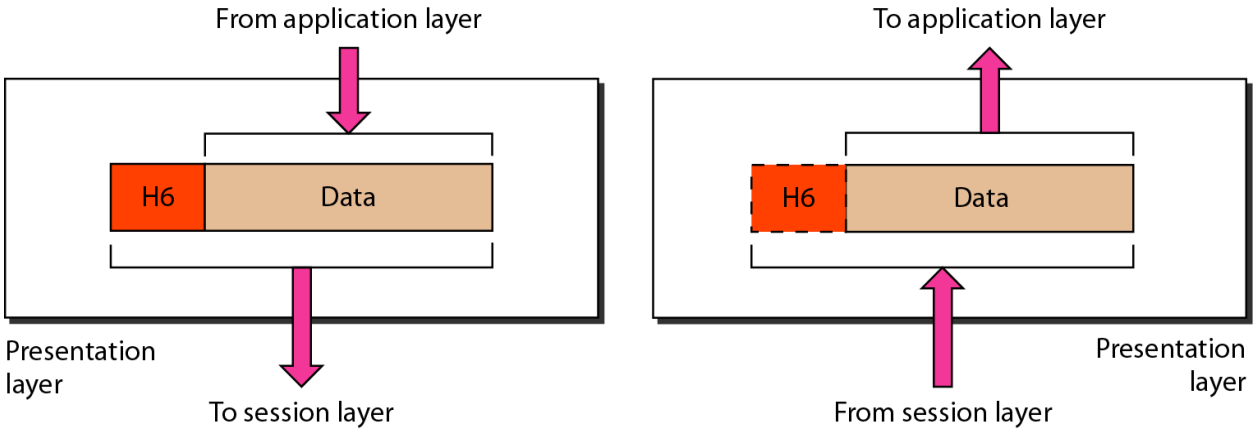
- The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers etc. The information must be changed to bits streams before being transmitted. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods.
- The presentation layer at the sender changes the information from its sender-dependent format into a common format.
- The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

**Encryption**

- Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network.
- Decryption reverses the original process to transform the message back to its original form. Encryption and Decryption is done for privacy of the sensitive information.

**Compression**

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




## Application Layer

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

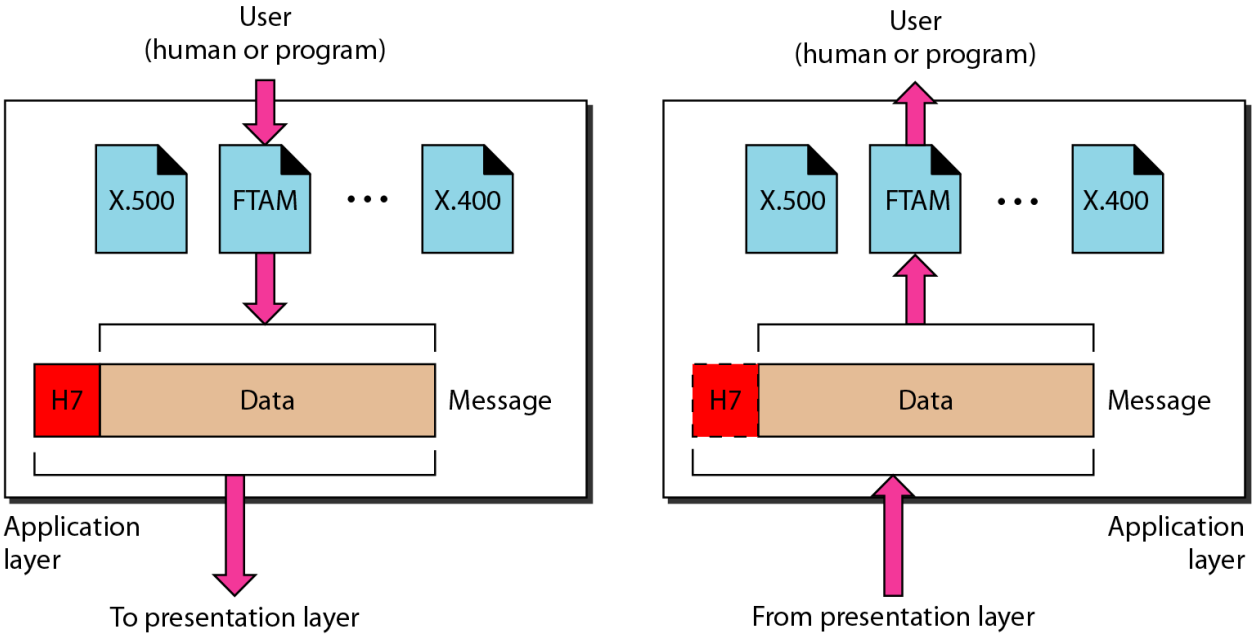
The **application layer** contains a variety of protocols that are commonly needed by users. The application layer enables the user to access the network.

Specific services provided by the application layer include the following:

- **A network virtual terminal** is a software version of a physical terminal,

and it allows a user to log on to a remote host.

- **File** transfer, access, and management in a remote host.
- **Mail services** such as email forwarding and mail storage.
- **Directory services** are an application provides distributed database sources and access for global information about various objects and services.

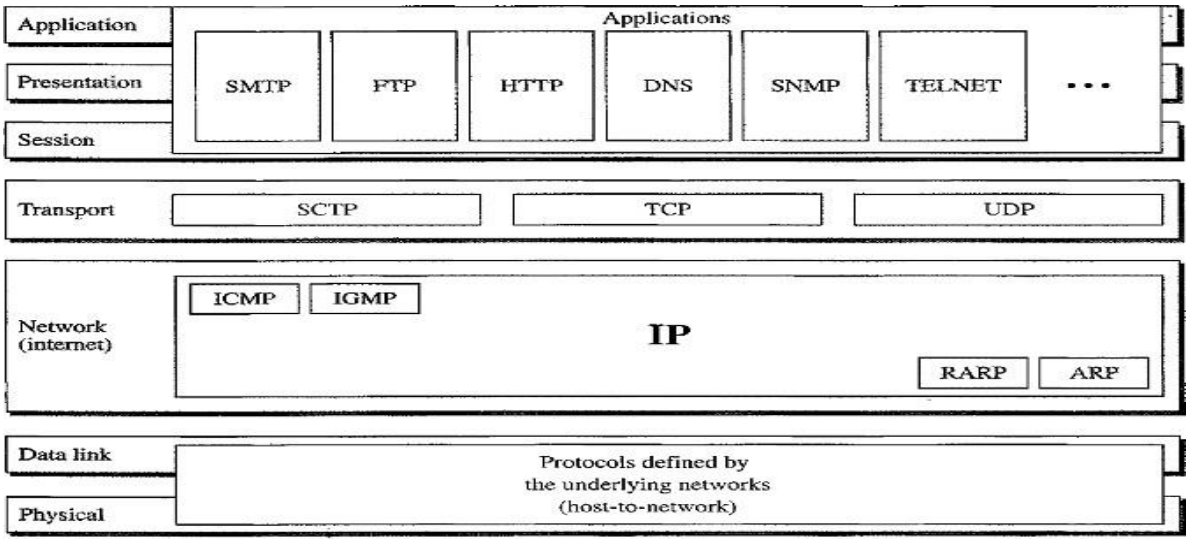


## TCP/IP PROTOCOL SUITE

The TCP/IP protocol suite was developed prior to the OSI model.

The original TCP/IP protocol suite was defined as having four layers:

1. Host-To-Network Layer
2. Internet Layer
3. Transport Layer
4. Application Layer



**Layers comparison in TCP/IP and OSI:**

- **Host-to-Network** layer is equivalent to the combination of the **Physical** and **Data link** layers.
- The **Internet Layer** is equivalent to the **Network layer**.
- The **Transport layer** is similar in both OSI and TCP/IP, except that in TCP/IP it will take care of part of the duties of the session layer.
- The **Application Layer** is roughly doing the job of the **Session, Presentation** and **Application** layers.

**Functionality in TCP/IP and OSI:**

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

**Host-to- Network Layer**

- At the Host-to-Network layer is a combination of Physical Layer and Data-link layer in OSI model.
- It is an interface between hosts and transmission links.
- **TCP/IP** does not define any specific protocol. It supports all the standard and proprietary protocols.
- A network in a TCP/IP internetwork can be a local-area network or a wide-area network.

**Internet Layer**(or) **Network Layer**

- In this layer *TCP/IP* supports the Internetworking Protocol (IP). The Internetworking Protocol (IP) is the transmission mechanism used by the TCP/IP protocols.
- It is an unreliable and connectionless protocol-a best-effort delivery service.
- The term *best effort* means that IP provides no error checking or tracking.
- The transmission is unreliable (i.e.) there is no guarantee for the data.
- IP transports data in packets called *datagrams,* each of which is transported separately.
- Datagrams can travel along different routes and can arrive out of sequence or be duplicated.
- IP does not keep track of the routes and has no facility for reordering datagrams once they arrive at their destination.

**IP uses four supporting protocols**

1. ARP (Address Resolution Protocol)
2. RARP(Reverse Address Resolution Protocol)
3. ICMP(Internet Control Message Protocol)
4. IGMP(Internet Group Message Protocol)

**Address Resolution Protocol (ARP)**

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

**Reverse Address Resolution Protocol (RARP)**

- RARP allows a host to discover its logical address when it knows only physical address.
- It is used when a computer is connected to a network for the first time or when a diskless computer is booted.

**Internet Control Message Protocol (ICMP)**

- ICMP is a mechanism used by hosts and gateways to send notification of datagram problems back to the sender.
- ICMP sends query and error reporting messages.

**Internet Group Message Protocol (IGMP)**

- IGMP is used to facilitate the simultaneous transmission of a message to a group of recipients.

**Transport Layer**

Transport layer in *TCP/IP* has three protocols:

1. **TCP** (*Transmission Control Protocol)*
2. **UDP**(*User Datagram Protocol)*
3. **SCTP**(Stream Control Transmission Protocol)

**Note:** UDP and TCP are transport level protocols responsible for delivery of a message from one device to another device, whereas IP is a host-to- host protocol meaning that it can deliver a packet from one physical device to another.

**Transmission Control Protocol**

- TCP provides full transport- layer services to applications. TCP is a reliable stream transport protocol.
- The term ***stream*** means connection-oriented: A connection must be established between both ends of a transmission before either can transmit data.
- At the sending side for each transmission TCP divides a stream of data into smaller units called *Segments*. Each segment includes a sequence number for reordering at the destination side. Segments are carried across the internet inside of IP datagrams.
- For every segment there is a corresponding acknowledgement to be sent from the destination to the source.
- At the receiving side TCP collects each datagram as it comes in and reorders the transmission based on sequence numbers.

**User Datagram Protocol**

- UDP is unreliable, connectionless protocols for applications that do not want TCP's sequencing or flow control and wish to provide their own.
- It is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from the upper layer.

- It is also widely used for client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery such as transmitting speech or video.

**Stream Control Transmission Protocol**

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

**Application Layer**

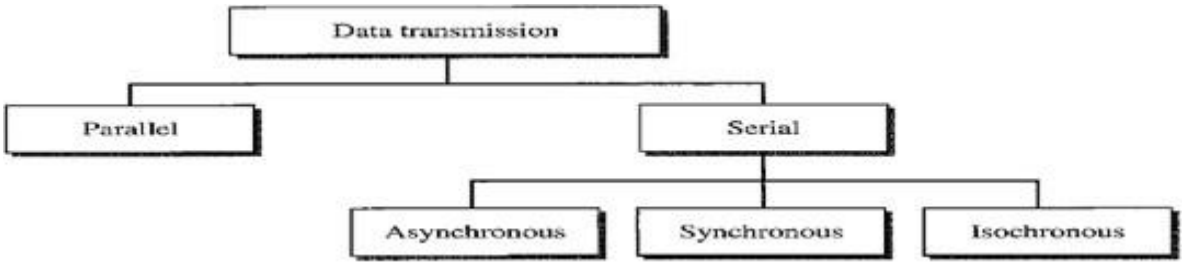
On top of the transport layer is the **application layer**. It contains all the higher-level protocols such as:

- **Telnet protocol** used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.
- **File Transfer Protocol (FTP)** used for file transfer.
- **Simple Mail Transfer Protocol (SMTP)** used for mail services.
- **Domain Name System (DNS)** used for mapping host names onto their network addresses.
- **Hyper Text Transfer Protocol (HTTP)** used for fetching pages on the World Wide Web (WWW).
- **Real-time Transport Protocol (RTP)** used for delivering real-time media such as voice or movies.

## TRANSMISSION MODES

Transmission modes are two types:

1. Parallel Transmission
2. Serial Transmission




### Parallel Transmission

Parallel Transmission is defined as sending n bits of data at a time instead of transmitting one bit at a time.

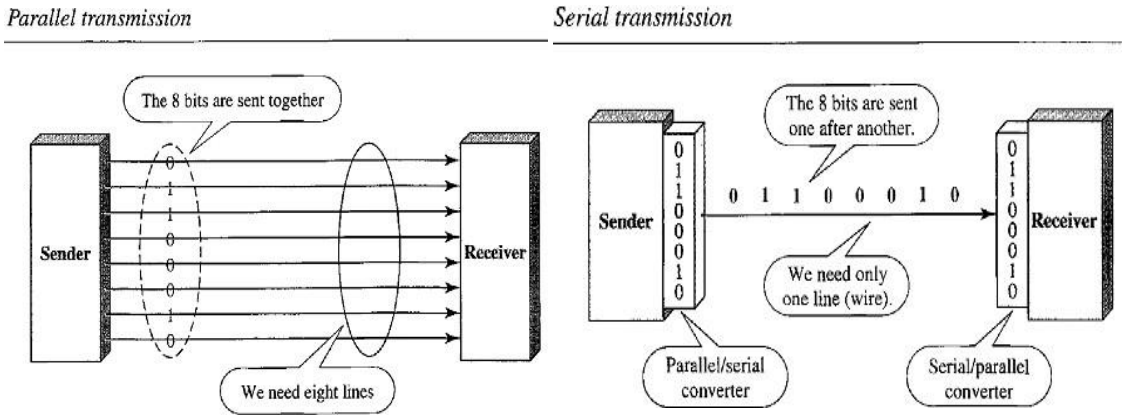
The mechanism for parallel transmission is a conceptually simple one: Use **n-wires** to send **n-bits** at one time.

**Advantage**: Speed of the transmission is increased.

**Disadvantage** : Cost of equipment is increased for this reason parallel transmission is usually limited to short distances.

### Serial Transmission

In serial transmission one bit follows another, so we need only one communication channel rather than **n channels** to transmit data between two communicating devices




**Advantage:** Reduces the cost transmission equipment because we need only one communication channel.

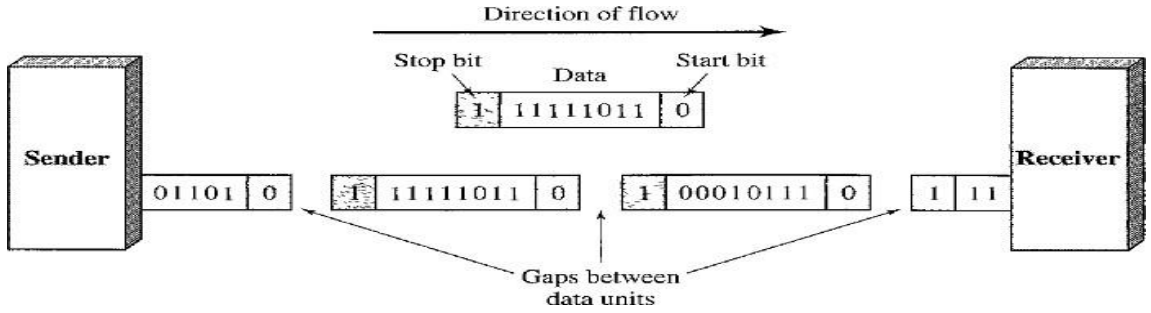
Since communication within devices is parallel, conversion devices are required at the interface between the sender and the line (parallel- to-serial) and between the line and the receiver (serial-to-parallel).

Serial transmission categorized into 3 types:

1. Asynchronous Transmission
2. Synchronous Transmission
3. Isochronous Transmission

## Asynchronous Transmission

- The timing of signal is not important in Asynchronous transmission. Information is received and translated by agreed upon patterns.
- As long as those patterns are followed, the receiving device can retrieve the information without regard to the order in which it is sent.
- Patterns are based on grouping the bit stream into bytes. Each group contains 8 bits is sent along the link as a unit.



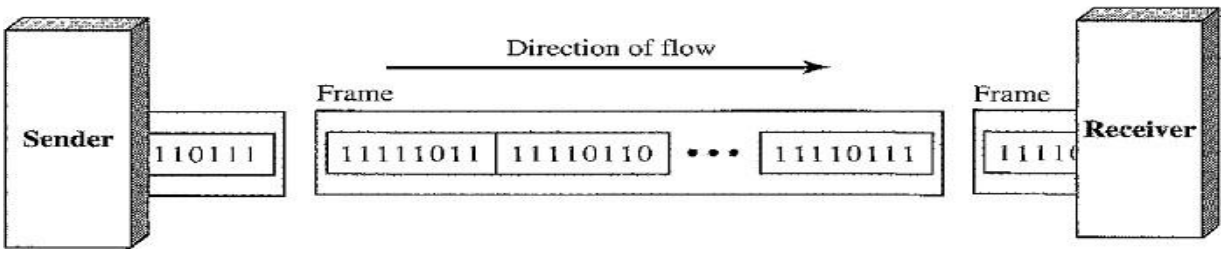

- In asynchronous transmission, we send one start bit (0) at the beginning and one or more stop bits (1's) at the end of each byte. There may be a gap between each byte.
- The start and stop bits are used because the sending system handles each group independently whenever the group is ready it will transmitted through the link.
- Without synchronization, the receiver cannot use timing to predict when the next group will arrive.
- To alert the receiver to the arrival of a new group the extra bits 0 and 1 are added.
- At the receiver side when the receiver detects a start bit, it sets a timer and begins counting bits as they come in. After *n* bits, the receiver looks for a stop bit. As soon as it detects the stop bit, it waits until it detects the next start bit
- **Start** and **Stop** bits and the **Gap** alert the receiver to the beginning and end of each byte and allow it to synchronize with data stream. This mechanism is called **Asynchronous.**
- The transmission is slow because of addition of start, stops and gaps between bit streams. Hence it is used for low speed communications.
- Example: The connection to the keyboard to the computer is application of Asynchronous transmission.
- Apart from slower transmission Asynchronous transmission is cheap and effective.

## Synchronous Transmission

In synchronous transmission, we send bits one after another without start or stop bits or gaps. It is the responsibility of the receiver to group the bits.

**That means:**

- The bit stream is combined into longer "**Frames**," which may contain multiple bytes.
- Each byte is introduced onto the transmission link without a gap between the byte and the next byte.
- It is left to the receiver to separate the bit stream into bytes for decoding purposes.
- Data are transmitted as an unbroken string of 1s and 0's, and the receiver separates that string into the bytes, or characters and receiver needs to reconstruct the information.




In synchronous transmission **Timing** plays very crucial role. When the information comes from sender, the receiving device **accurately count the bits** and group them into 8 bits because we don't have any extra bits to identify starting and ending of byte. This process is called **Byte Synchronization**.

**Advantage**: Speed of the transmission is increased as compared to Asynchronous transmission because there are no extra bits to be add or remove at the sender side and receiver side respectively.

It is useful for **High Speed Application** such as transmission of data from one computer to another computer.

**Note:**

1. Byte Synchronization is accomplished at Receiver side.
2. Although there is no gap between characters in synchronous serial transmission, there may be uneven gaps between frames.

**Isochronous Transmission**

- The isochronous transmission guarantees that the data arrive at a fixed rate.
- In real- time audio and video, in which synchronous transmission fails such as uneven delays between frames, are not acceptable.
- **For example**, TV images are broadcast at the rate of 30 images per second; they must be viewed at the same rate. If each image is sent by using one or more frames, there should be no delays between frames.
- For this type of application, synchronization between characters is not enough; the entire stream of bits must be synchronized.

# Multiplexing

**Multiplexing** is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

As data and telecommunications use increases the data traffic is also increases.

We can accommodate this increase by continuously adding the individual links each time a new channel is needed, or we can install higher-bandwidth links and use each to carry multiple signals.

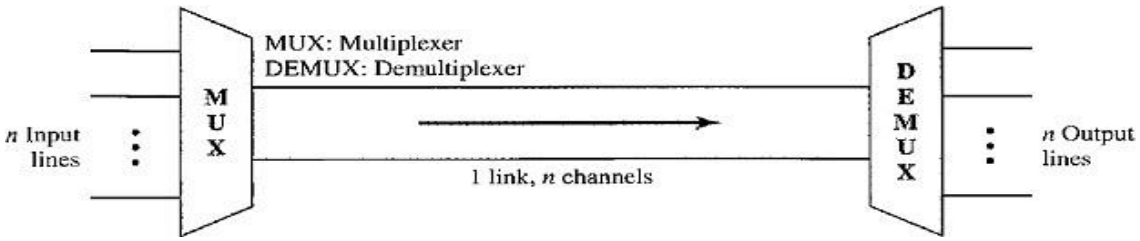


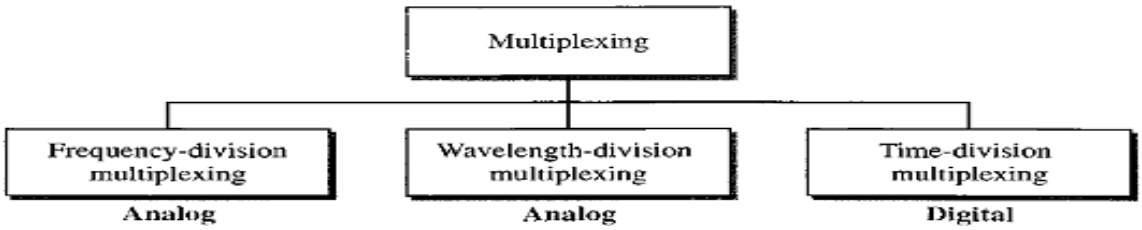

Fig: Dividing the link into channels

In a multiplexed system, *n* lines share the bandwidth of one link.

- **Link** refers to the physical path.
- **Channel** refers to the portion of a link that carries a transmission between a given pair of lines.
- The lines on the left direct their transmission streams to a **Multiplexer (MUX)**, which combines them into a single stream (many-to-one).
- At the receiving end, that stream is fed into a **Demultiplexer (DEMUX)**, which separates the stream back into its component transmissions (one-to-many) and directs them to their corresponding lines.

Multiplexing is categorized into 3 types:

1. Frequency Division Multiplexing
2. Wavelength Division Multiplexing
3. Time Division Multiplexing




## Frequency Division Multiplexing(FDM)

FDM is an analog multiplexing technique that combines analog signals.

**That means:**

- FDM is an analog technique that can be applied when:
  **Bandwidth of link (in Hz) >= Combined bandwidth of the signal 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.

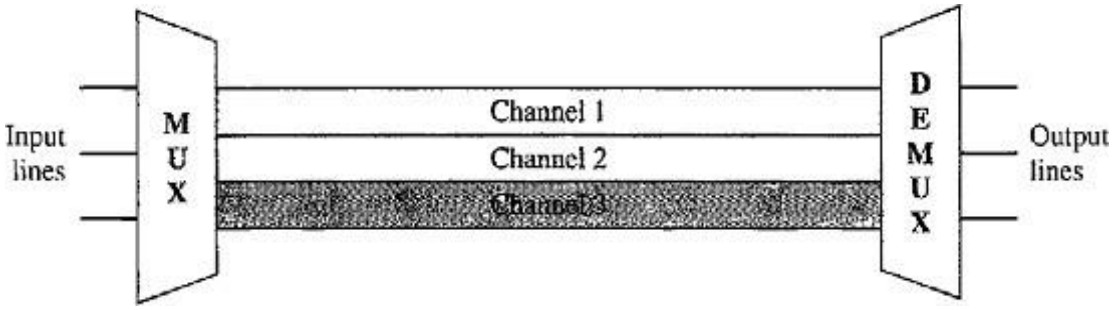



Fig: Frequency Division Multiplexing

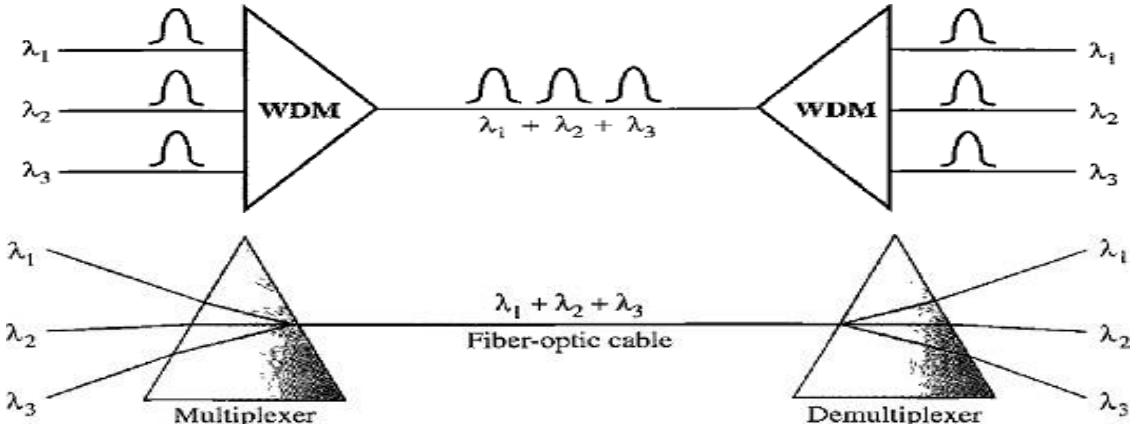
In the above figure, the transmission path is divided into three parts, each representing a channel that carries one transmission.

**Multiplexing Process**

- Each source generates a signal of a similar frequency range.
- Inside the multiplexer, these similar signals modulate different carrier frequencies (f1, f2, f3).
- The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.

**Wavelength-Division Multiplexing (WDM)**

- 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 **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.




- Very narrow bands of light from different sources are combined to make a wider band of light. At the receiver, the signals are separated by the demultiplexer.

- A demultiplexer can be made to divide wider band of frequencies by decomposing the light beams into narrow band frequencies.

**Advantages**: High Speed and High frequency, uses narrow bands of light sources.

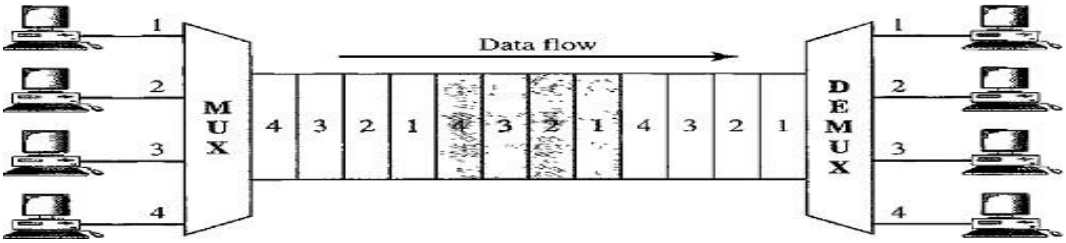
**Disadvantages**: Expensive than FDM.

## Time-Division Multiplexing (TDM)

TDM is a digital multiplexing technique for combining several low-rate channels into one high-rate channel. Digital data from different sources are combined into one timeshared link

(i.e.) The **data rate** capacity of transmission medium >= The **data rate** required by sending and receiving devices.

TDM is a digital process that allows several connections to share the high bandwidth of a link. Each connection occupies a portion of time in the link.

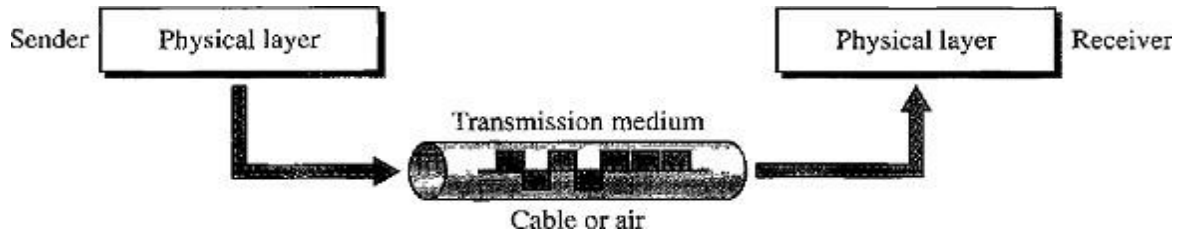



In the above figure all the data in a message from source 1 always go to one specific destination either of 1, 2, 3, or 4. The delivery is fixed and unvarying.

## TRANSMISSION MEDIA

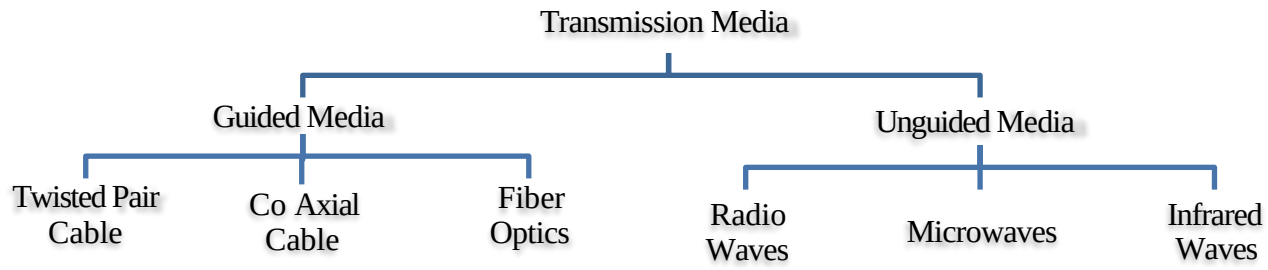
Transmission media are actually located below the physical layer and are directly controlled by the physical layer.

A transmission **medium** can be broadly defined as anything that can carry information from a source to a destination. In data communications the information is usually a signal.




Transmission media can be categorized into following ways:

- **Guided or Wired Media**: Twisted pair cable, Coaxial cable, Fiber Optic cable.
- **Unguided or Wireless Media**: Radio Waves, Micro waves, Infrared Waves.



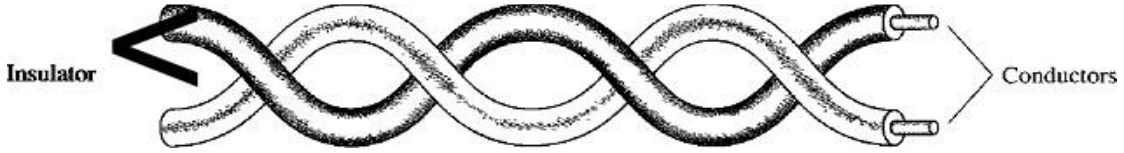

### Guided or Wired Media

A signal traveling along this media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light.

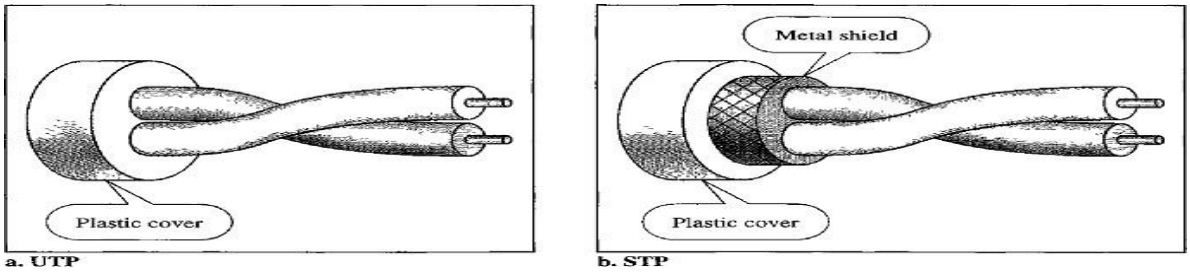
**Twisted-Pair Cable**

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

One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference.




The signal sent by the sender on one of the wires, interference (noise) and crosstalk mayaffect both wires and create unwanted signals.




a. UTP b. STP

**STP v/s UTP**

**Shielded Twisted Pair** (STP) cable has a **metal foil** or braided mesh covering that encases each pair of insulated conductors. A twisted-pair cable can pass a wide range of frequencies. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive.

**Unshielded Twisted pair** (UTP) cables don't have the metal foil covering the cables. The most common UTP connector is RJ45 (Registered Jack45).


**Applications**

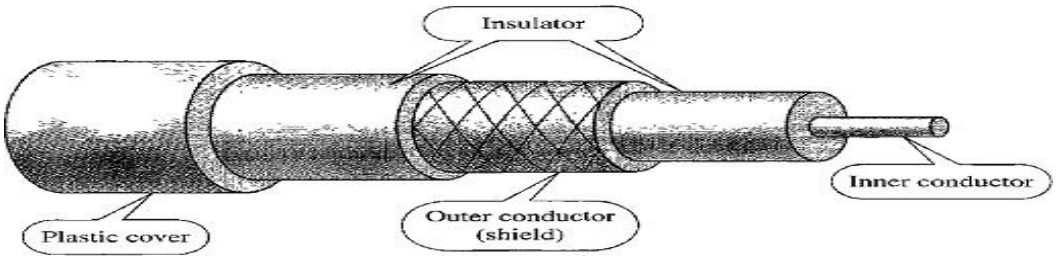
- Twisted-pair cables are used in telephone lines to provide voice and data channels. Most widely used in Internet connections.
- The DSL lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables.

Note: When there is an electric signal interference UTP signal performance is degraded.

Hence we use STP, the shield protects from interference of electric signals.

**Coaxial Cable (Coax)**

Coaxial cable carries signals of higher frequency ranges than those in twisted pair cable.



- Coaxial cable has a central core conductor of copper wire enclosed in an insulating sheath.
- Insulating sheath encased in an outer conductor of metal foil.
- The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit.
- This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover.

Coaxial cables are categorized by their Radio Government (RG) ratings. Each RG number denotes a unique set of physical specifications.

| *Category* | *Impedance* | *Use* |
|---|---|---|
| RG-59 | 75 Ω | Cable TV |
| RG-58 | 50 Ω | Thin Ethernet |
| RG-11 | 50 Ω | Thick Ethernet |

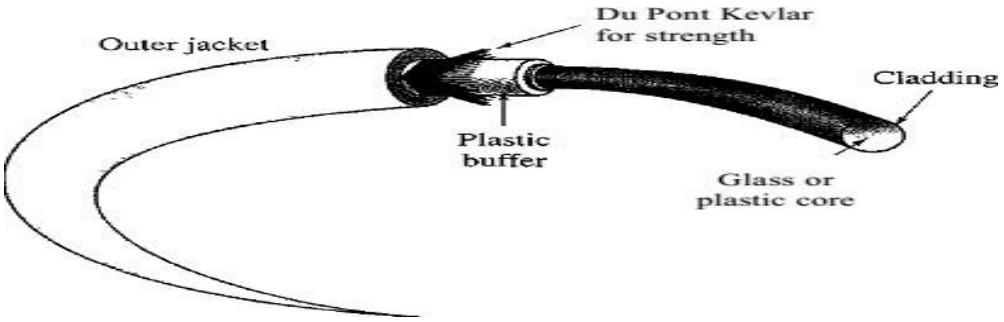
Although coaxial cable has a much higher bandwidth, the signal weakens rapidly and requires the frequent use of repeaters.

**Applications**

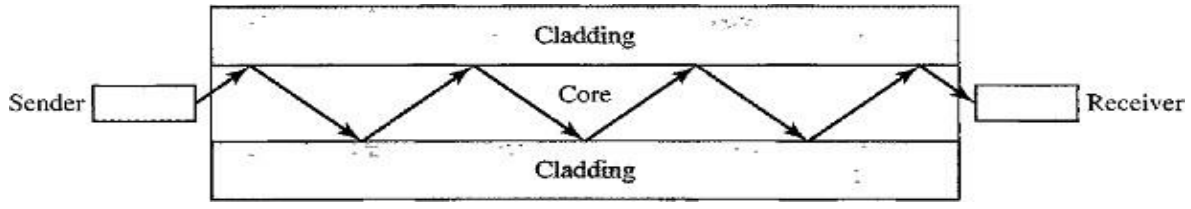
Coaxial cable was widely used in analog telephone networks, digital telephone networks, Cable TV networks, Ethernet LAN.

**Fiber-Optic Cable**

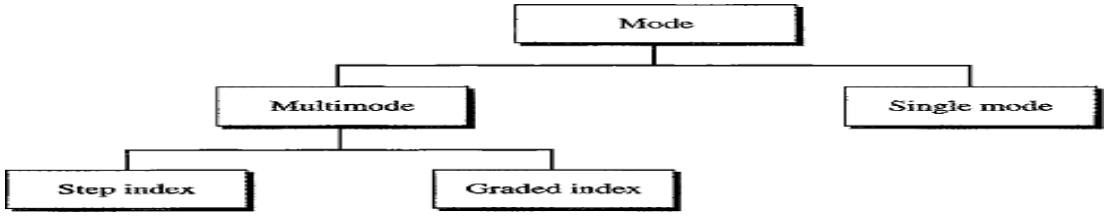
- A fiber-optic cable is made of glass or plastic and transmits signals in the form of light.
- The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable.
- Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.




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



**Propagation Modes**




**Multimode Propagation**

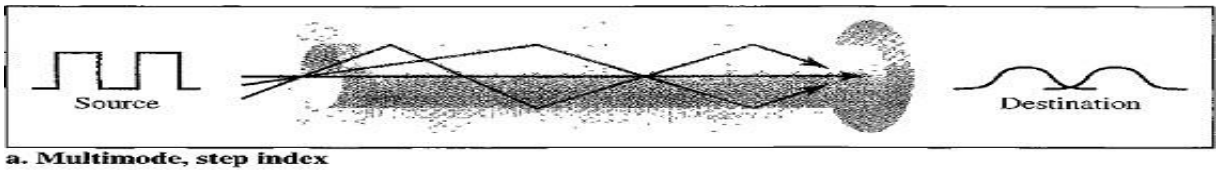
In this mode multiple beams from a light source move through the core in different paths.

**Multimode Step-Index Fiber:**

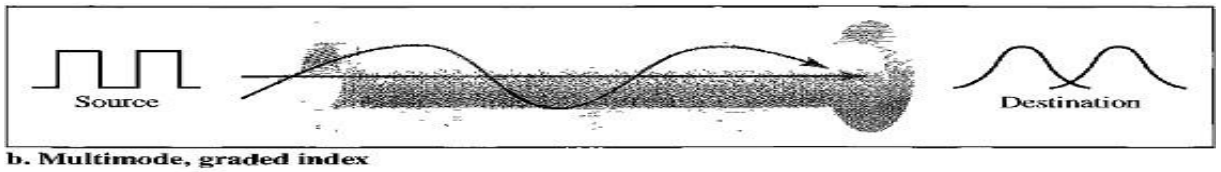
- The density of the core remains constant from the center to the edges.
- A beam of light moves through this constant density in a straight line until it reaches the interface of the core and the cladding.
- At the interface, there is an abrupt change due to a lower density; this alters the angle of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

**Multimode Graded-Index Fiber:**

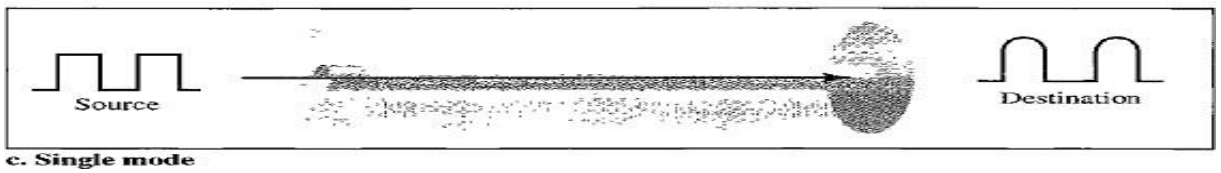
- It decreases the distortion of the signal through the cable.
- The word *index* here refers to the index of refraction.
- The index of refraction is related to density. A graded- index fiber is one with varying densities.
- Density is highest at the center of the core and decreases gradually to its lowest at the edge.




a. Multimode, step index




b. Multimode, graded index




c. Single mode

**Single-Mode Fiber:**

- It uses step- index fiber and a highly focused source of light that limits beams to a small range of angles close to the horizontal.
- The single mode fiber is manufactured with a much smaller diameter than that of multimode fiber, and with substantially lower density (index of refraction).

- The decrease in density results in a critical angle that is close enough to 90° to make the propagation of beams almost horizontal.
- In this case, propagation of different beams is almost identical, and delays are negligible.
- All the beams arrive at the destination "together" and can be recombined with little distortion to the signal.

**Fiber Optic Cable Connectors**

- The **subscriber channel** (SC) **connector** is used for cable TV. It uses a push/pull locking system.
- The **straight-tip** (ST) **connector** is used for connecting cable to networking devices.

**Performance:** The performance is such that we need 10 times less repeaters when we use fiber-optic cable.

**Application:** Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective.

**Advantages**

Fiber-optic cable has several advantages over metallic cable Twisted pair or coaxial.

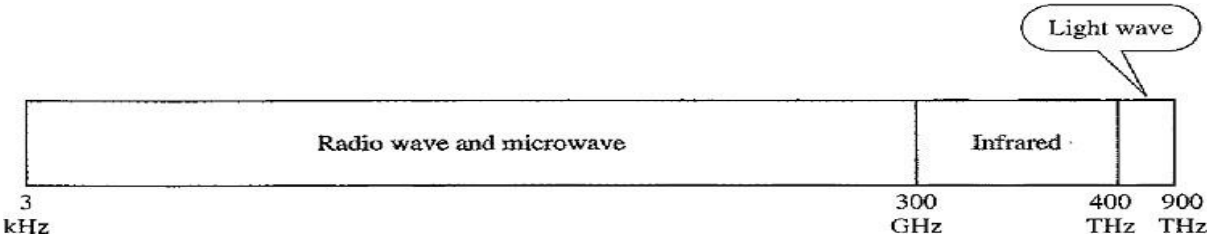
- **Higher bandwidth**. Fiber-optic cable can support higher bandwidths than either twisted- pair or coaxial cable.
- **Less signal attenuation**. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.
- **Immunity to electromagnetic interference** .Electromagnetic noise cannot affect fiber- optic cables.
- **Resistance:** Glass is more resistant to corrosive materials than copper.
- **Light weight.** Fiber-optic cables are much lighter than copper cables.
- **Greater immunity to tapping**: Fiber-optic cables are more immune to tapping than copper cables.

**Disadvantages**

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

## UNGUIDED MEDIA(or) WIRELESS COMMUNICATION

Unguided media transport **Electromagnetic Waves** without using a physical conductor. This type of communication is often referred to as wireless communication. Electromagnetic spectrum ranging from **3 kHz to 900 THz** used for wireless communication.

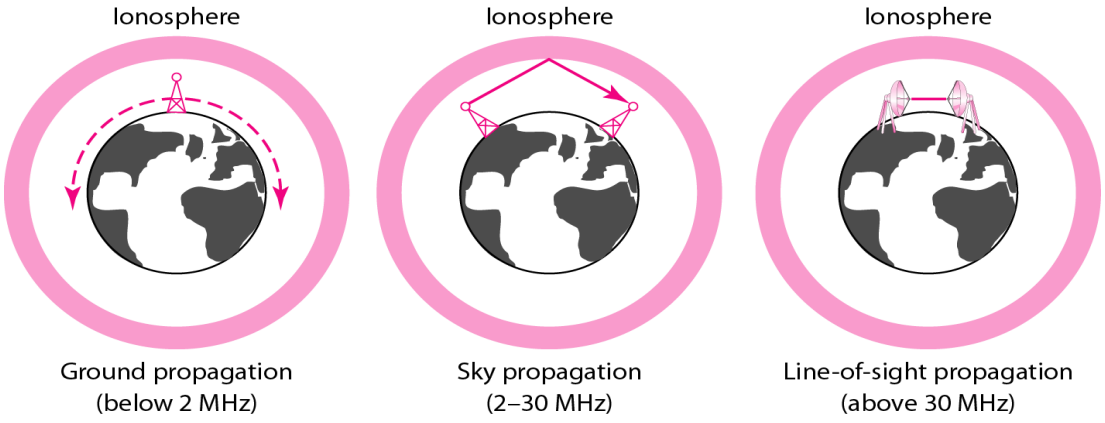



Categories of Wireless Communication:

- Radio Waves ( 3kHz –1GHz)
- Microwaves ( 1GHz- 300 GHz)
- Infrared Waves (300 GHz - 400 THz)

  Propagation Methods:

  - In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth.
  - In sky propagation, higher-frequency radio waves radiate upward into the ionosphere where they are reflected to earth. This type of transmission allows for greater distances with lower output power.
  - In line-or-sight propagation, very high-frequency signals are transmitted in straight lines directly from antenna to antenna.




## Radio Waves

- Radio waves ranges between 3 kHz and 1 GHz. Radio waves are Omni-directional.
- When an antenna transmits radio waves, they are propagated in all directions. Hence the sending and receiving devices don't have to be aligned.
- A sending antenna sends waves that can be received by any receiving antenna.
- Radio waves can travel long distances, hence it is used in long distance AM Radio broadcasting.
- Radio waves of low and medium frequencies can penetrate walls.

### Disadvantage

- The Omni-directional property has a **disadvantage**; the radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band.

- Radio waves leads to low data rate for digital communication.

**Applications**

- Radio waves are used in Multicasting applications such as AM Radio and FM radio, Television, Maritime Radio, Cordless Phones, and Paging.

**Micro waves**

- Electromagnetic waves having frequencies between 1GHz and 300 GHz are called microwaves.
- Microwaves are unidirectional. When an antenna transmits microwave waves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned.
- Microwaves need unidirectional antennas that send out signals in one direction.Two types of antennas are used for microwave communications: the parabolic dish and the horn.
- Microwave propagation is line-of-sight. Repeaters are often needed for long distance communication.
- Higher data rates are possible due to assigning of wider sub-bands.

**Advantage**

The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas.

**Disadvantage**

Very high- frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.

**Applications**

Microwaves used in Uni-casting communication between sender and receiver such as cellular phones, satellite networks and wireless LANs.

## Infrared Waves

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication upto few meters.

**Advantages**

Infrared waves having high frequencies cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room.

**Disadvantage**

- We cannot use Infrared waves for long range communication.
- We cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

**Applications**

- Due to its wide bandwidth, it can be used to transmit digital data at high data rate.
- It can be used in Communication between devices such as keyboards, mice, PCs, and printers

# Data Link Layer

The responsibility of the **Physical Layer** is to transmit the unstructured raw bit stream over a physical medium.

The responsibility of **Data-Link Layer** is to transforming raw transmission facility into a **Link** responsible for node-to-node communication (hop-to-hop communication).

## Responsibilities of the Data Link Layer include:

1. Framing
2. Physical Addressing
3. Flow control
4. Error control
5. Media Access Control.

## Framing

The data link layer divides the stream of bits received from the network layer into manageable data units called frames. In simple terms data link layer is responsible for moving frames from one node to another node.

## Physical Addressing

The data link layer adds a header to the frame to define the addresses of the sender and receiver of the frame.

## Flow Control

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

## Error Control

The data link layer also adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged, duplicate, or lost frames.

## Media 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.

## Framing

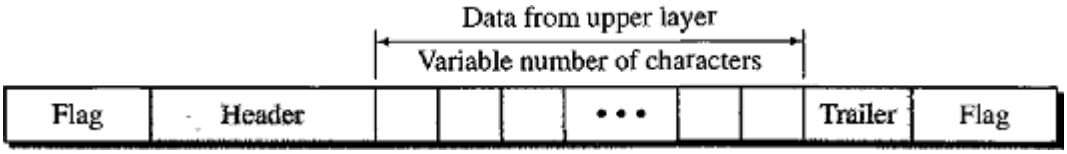
- Framing in the data link layer is breaking up the bit stream into frames.
- Framing can be done in 2ways:
- **Fixed size framing:** The size of the frame is fixed for all the frames. There is no need to define the boundaries of a frame.
- **Variable size framing:** In variable-size framing, we need a way to define the end of the frame and the beginning of the next frame.

There are 2 approaches are used for variable size framing:

1. Character Stuffing(A Character-Oriented Approach)
2. Bit Stuffing(A Bit-Oriented Approach)

## Character Stuffing/Byte Stuffing

In a character stuffing, data to be carried are 8-bit characters from a coding system such as ASCII. The Frame format in Character Stuffing is given below:




Character Stuffing uses: Header, Trailer and a Flag.

- **Header** carries the source and destination addresses and other control information.
- **Trailer** carries error detection or error correction redundant bits, are also multiples of 8 bits.
- To separate one frame from the next, an 8-bit (I-byte) flag is added at the beginning and the end of a frame. The flag signals receiver either start or end of a frame.

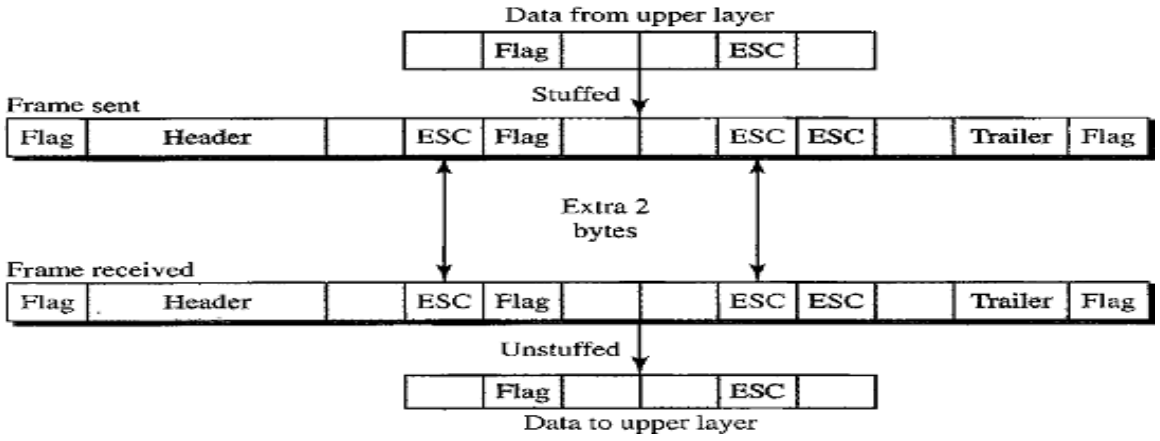
### Disadvantages of Character Stuffing

- Character oriented framing is useful for text transfer not useful for audio video etc.
- Any pattern used for the flag could also be part of the information.
- If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame and the treats then exit bit as new frame.

To fix this problem a **Byte Stuffing** strategy is introduced.

- In byte 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 called Escape character (ESC).
- 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.

Figure shows the byte stuffing and Unstuffing:



## Problems with Byte Stuffing

- If the text contains one or more escape characters followed by a flag, the receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame.

## Solution

- To solve this problem, the escape characters that are part of the text must also be marked by another escape character.

## Disadvantages of character/Byte stuffing Procedure

- The universal coding systems (Unicode) in use today have16-bit and 32-bit characters that conflict with 8-bit characters.
- Character stuffing deals with 8-bit characters but today's systems using 16 bits, 32 bits and 64 bit characters hence there will be conflict.

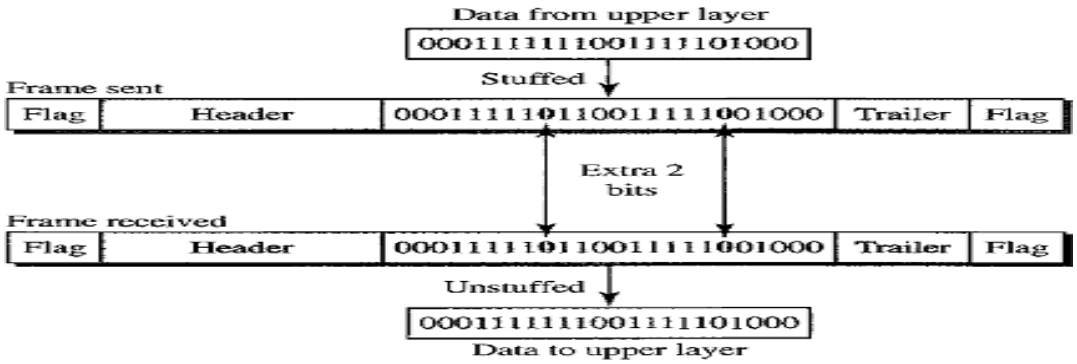
The solution for this problem is using ***Bit Oriented Approach.***

## Bit stuffing

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.
- In addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other.
- Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame is give below figure:

- In bit stuffing, if a 0 and five consecutive1-bits are encountered, an extra0 is added.
- This extra stuffed bit is eventually removed from the data by the receiver.

Note: the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. (i.e.) when 0**11111**00 is a part of the data, then also we have to add "0" after five 1"s .Hence the data will be 0**11111**000




## Advantages of Bit Stuffing

If the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

**Error** is a condition when the receiver's information does not match the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits traveling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error Detection:
To prevent such errors, error-detection codes are added as extra data to digital messages. This helps in detecting any errors that may have occurred during message transmission.

1. **Checksum**

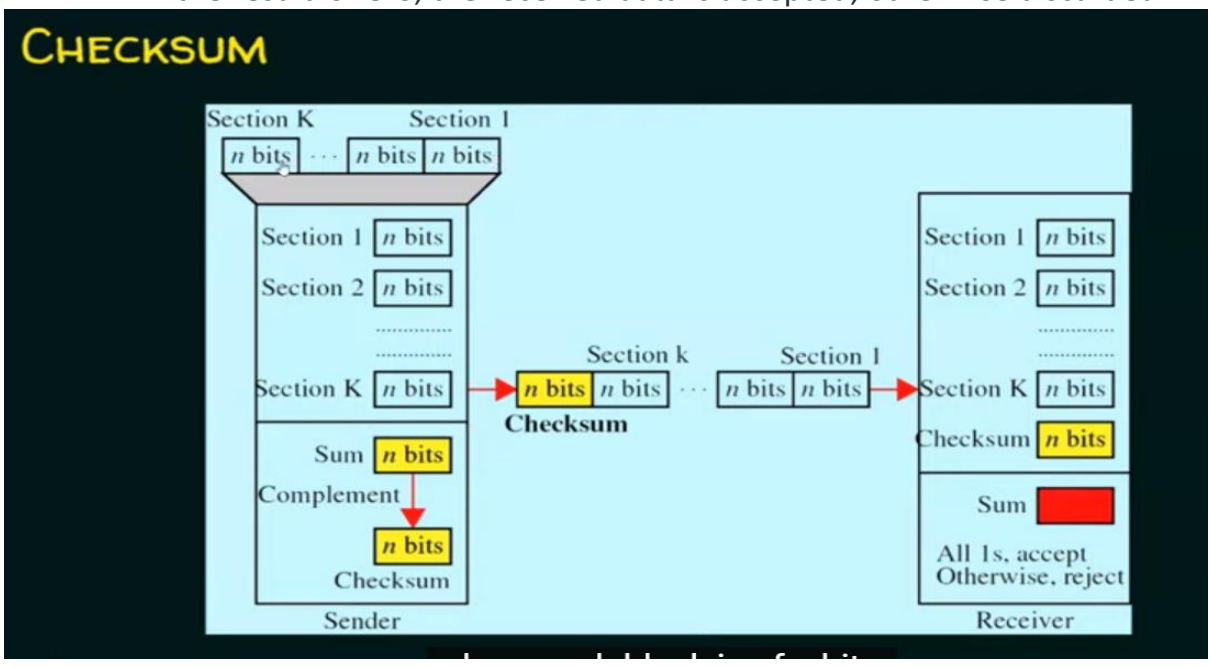
Checksum error detection is a method used to identify errors in transmitted data. The process involves dividing the data into equally sized segments and using a 1's complement to calculate the sum of these segments. The calculated sum is then sent along with the data to the receiver. At the receiver's end, the same process is repeated and if all zeroes are obtained in the sum, it means that the data is correct.

**Checksum – Operation at Sender's Side**

- Firstly, the data is divided into k segments each of m bits.
- On the sender's end, the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.

**Checksum – Operation at Receiver's Side**

- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.




Example:

The first step is to perform the bit addition of the given data bits at the sender side.

**Sender Side:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| | | | | | | **1** | **0** |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

Note: The extra carry bits are added to the summation result.

2. Perform the 1's Complement for the bit addition result, thus obtaining the checksum value.

**Sender Side:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

**1's Complement**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Checksum** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0** |

3. Integrate the checksum value and the original data bit and begin the data transmission to the receiver.

| | | | | |
|---|---|---|---|---|
| **11011010** | 10011001 | 11100010 | 00100100 | 10000100 |

4. The receiver side will begin the Checksum Checker method, repeat the bit addition, and perform the 1's complement.

**Receiver Side:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0** |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| | | | | | | **1** | **0** |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

5. If the complement result is 0, the data received is correct and without any error.

**Receiver Side:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**1's Complement**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Checksum** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**Result:** No error in the data received from the sender side.

## Cyclic Redundancy Check (CRC)

- Unlike the checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of the data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

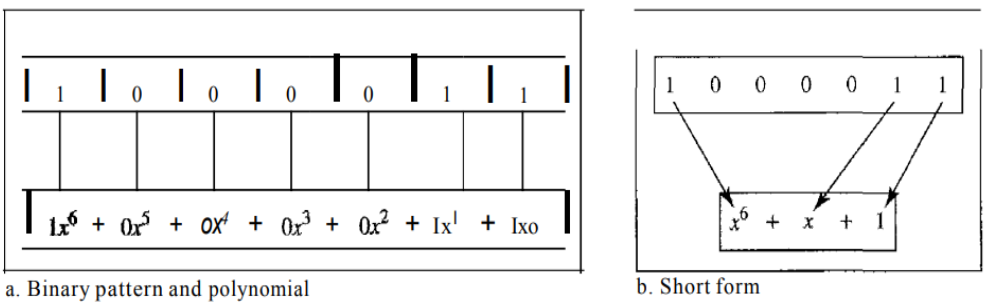
**Design:**

- **In** the encoder, the data word has **k bits** and the codeword has **n bits**.
- The size of the data word is augmented by adding **(n–k)number of 0's** 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+1** predefined and agreed by both the sender and receiver.
- The generator divides the augmented data word by the divisor**(modulo-2 division)**.
- **The quotient of the division is discarded;**
- **The remainder ($r_2r_1$ $r_0$)is appended to the dataword to create the codeword.**

**Polynomials:**

- A pattern of Os and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.

**Figure 10.21** *A polynomial to represent a binary word*

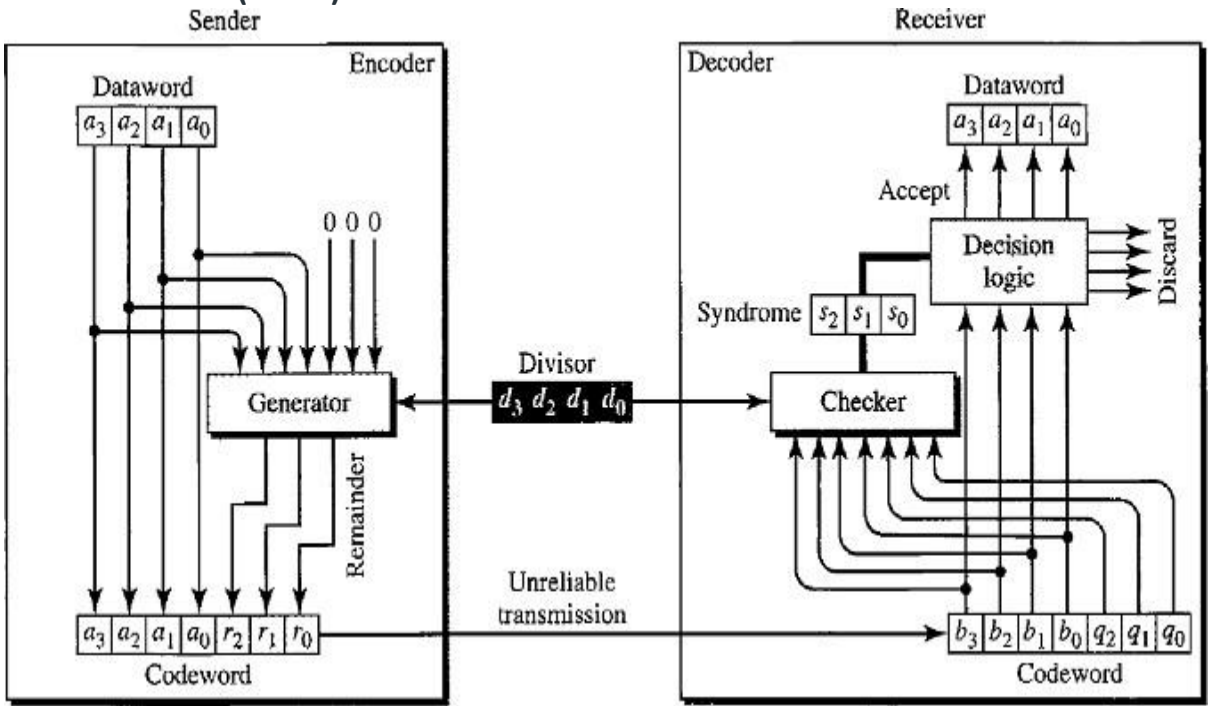



a. Binary pattern and polynomial

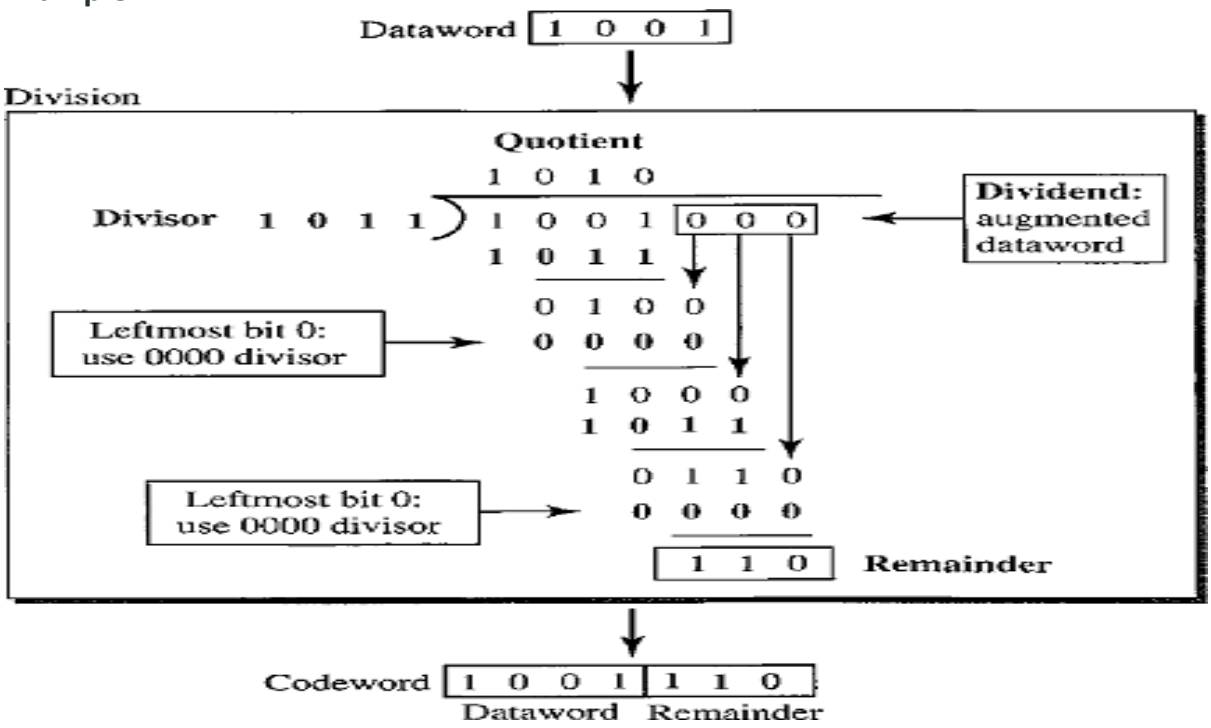
b. Short form

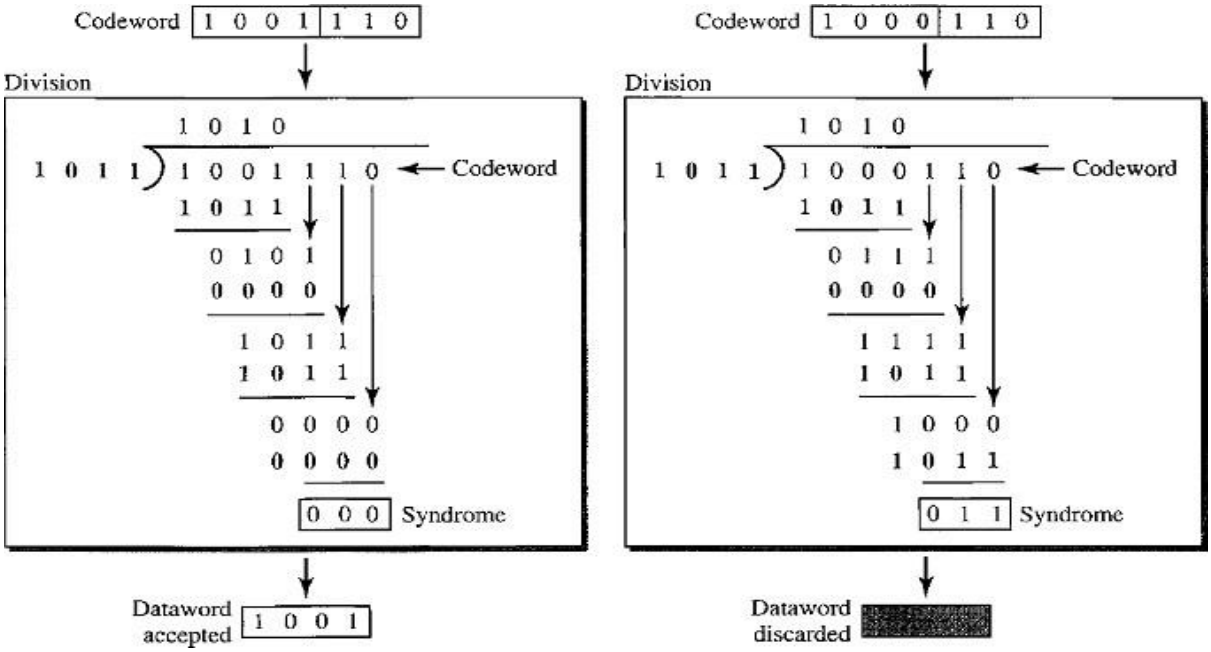
**Example:**

- **Let us take k=4 bits n=7 bits**
- **Appended Dataword Size = (n-k) = 3.**
- **Divisor Size=(n-k+1)=4.**




**Example:**






**Example 2:**

original message
1 0 1 0 0 0 0

@ means X-OR

Generator polynomial
$x^3+1$
$1.x^3+0.x^2+0.x^1+1.x^0$
CRC generator
1 0 0 1 4-bit

If CRC generator is of n bit then append (n-1) zeros in the end of original message

Sender

1 0 0 1 | 1 0 1 0 0 0 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0 0 0 0
@1 0 0 1
0 1 0 1 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0
@1 0 0 1
0 1 0 1 0
@1 0 0 1
0 0 1 1

Message to be transmitted
1 0 1 0 0 0 0 0 0 0
+ 0 1 1
1 0 1 0 0 0 0 0 1 1

1 0 0 1 | 1 0 1 0 0 0 0 0 1 1
@1 0 0 1
0 0 1 1 0 0 0 0 1 1
@1 0 0 1
0 1 0 1 0 0 1 1
@1 0 0 1
0 0 1 1 0 1 1
@1 0 0 1
0 1 0 0 1
@1 0 0 1
0 0 0 0

Receiver

Zero means data is accepted

**Error Correction:**

**Hamming code** is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver.

**Redundant bits –** Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

```
2^r ≥ m + r + 1

 where, r = redundant bit, m = data bit
```

m=7

2^4 ≥ 7 + 4 + 1 Thus, the number of redundant bits= 4 **Parity bits.**

There are two types of parity bits:

1. **Even parity bit:** In the case of even parity, for a given set of bits, the number of 1's are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of 1's in a given set of bits is already even, the parity bit's value is 0.
2. **Odd Parity bit –** In the case of odd parity, for a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1's an odd number. If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

**General Algorithm of Hamming code:** Hamming Code is simply the use of extra parity bits to allow the identification of an error.

1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form. **a.** Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc). **b.** Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc). **c.** Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc). **d.** Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47,

etc). **e.** In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

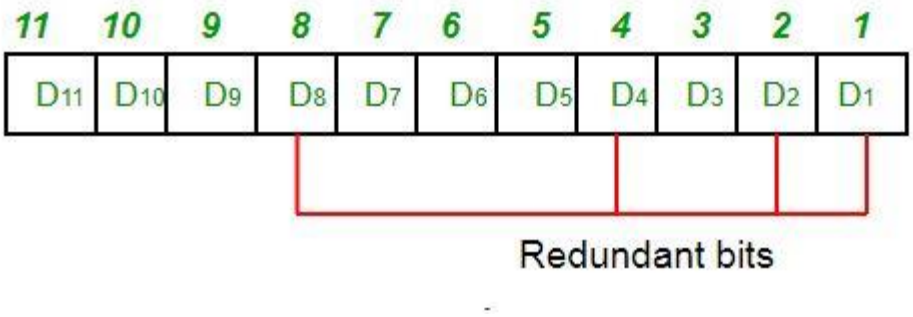
| Position | R8 | R4 | R2 | R1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |

R1 -> 1,3,5,7,9,11
R2 -> 2,3,6,7,10,11
R3 -> 4,5,6,7
R4 -> 8,9,10,11

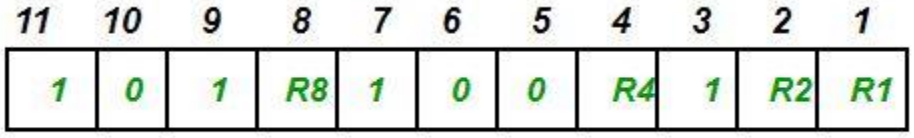
**Determining the position of redundant bits –** These redundancy bits are placed at positions that correspond to the power of 2.
As in the above example:

- The number of data bits = 7

- The number of redundant bits = 4
- The total number of bits = 11
- The redundant bits are placed at positions corresponding to power of 2- 1, 2, 4, and 8

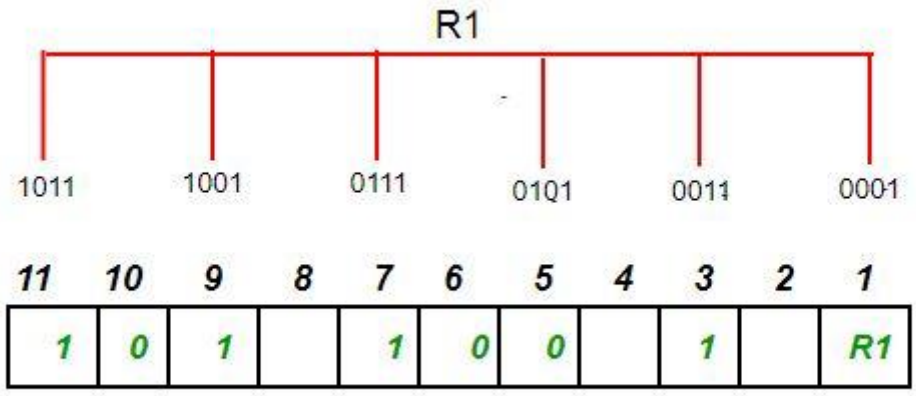



- Suppose the data to be transmitted is 1011001, the bits will be placed as follows:

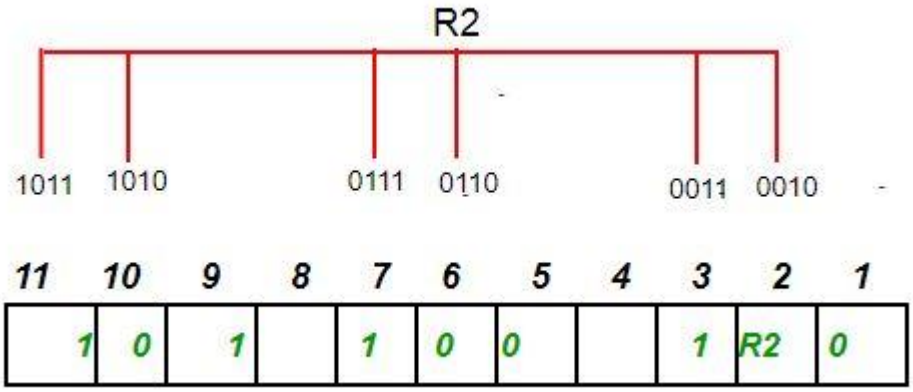



**Determining the Parity bits:**

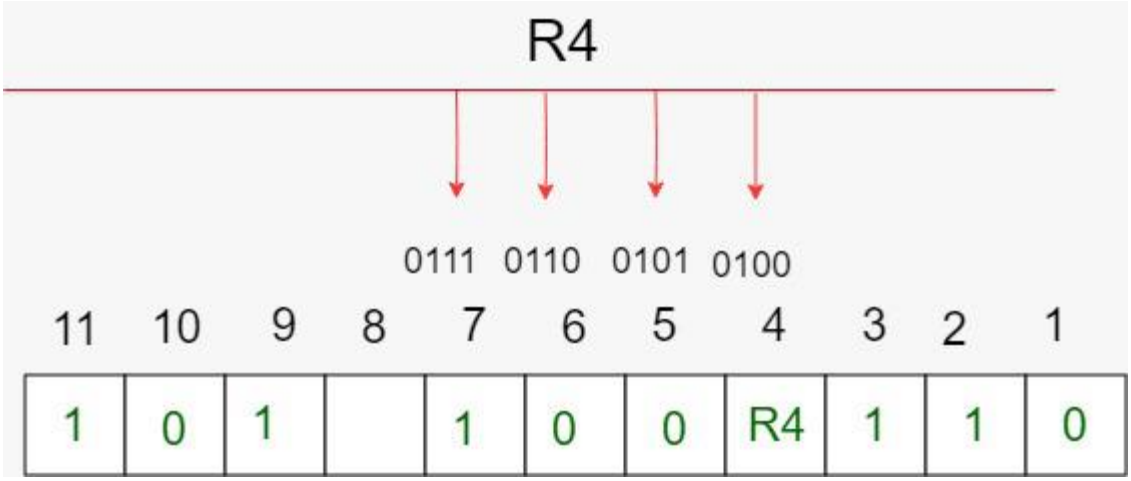
- R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position. R1: bits 1, 3, 5, 7, 9, 11



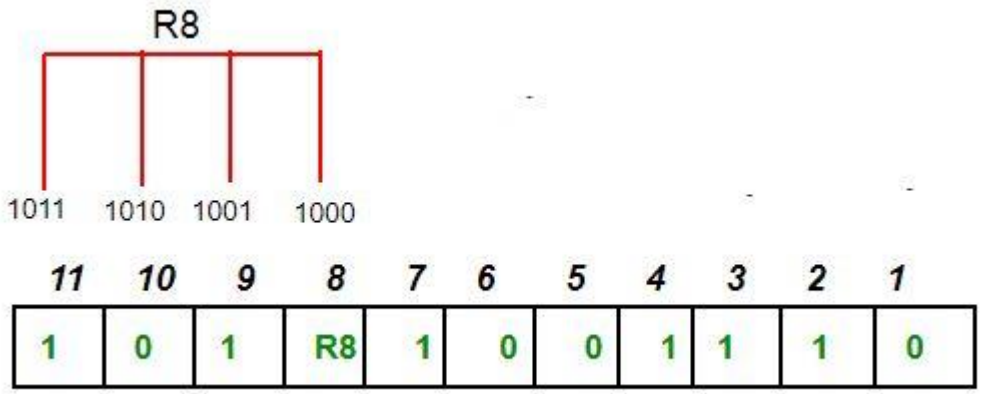

- To find the redundant bit R1, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit's value) = 0
- R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit. R2: bits 2,3,6,7,10,11



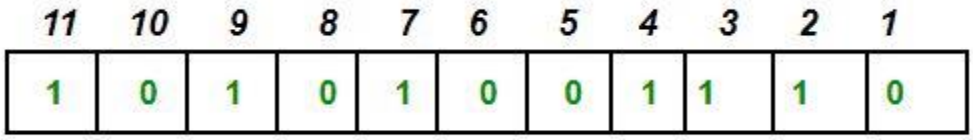

- To find the redundant bit R2, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R2 is odd the value of R2(parity bit's value)=1
- R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit. R4: bits 4, 5, 6, 7



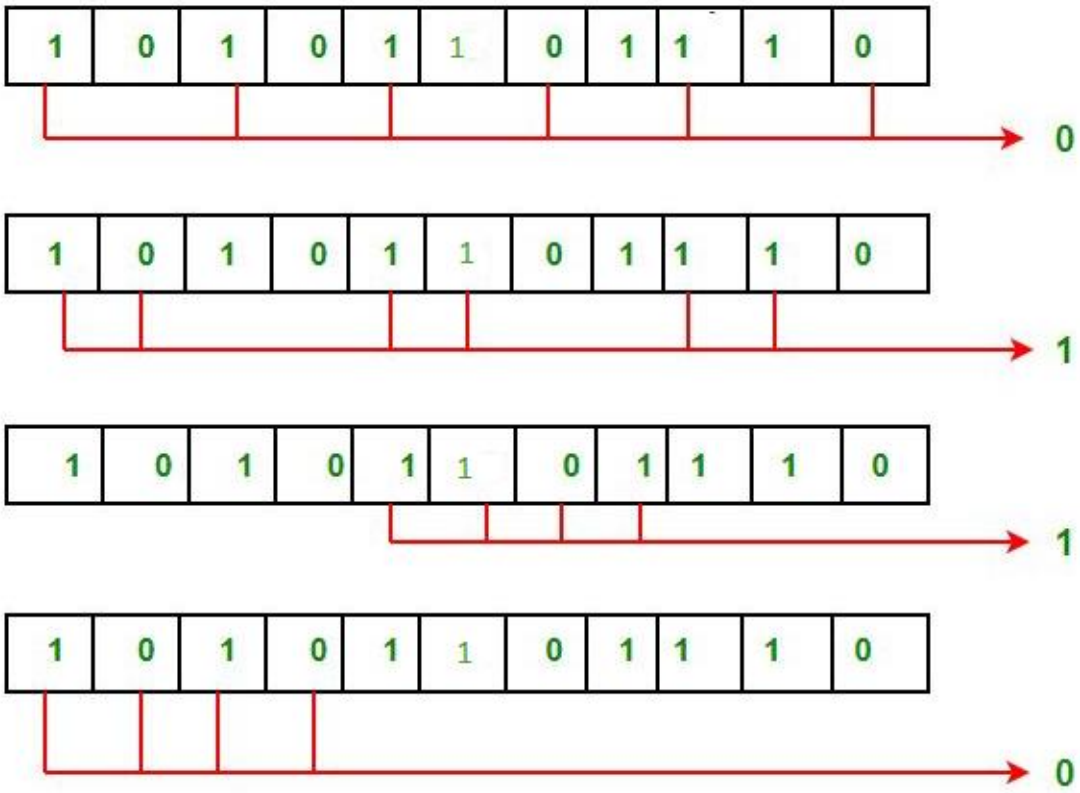

1. To find the redundant bit R4, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R4 is odd the value of R4(parity bit's value) = 1
2. R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit. R8: bit
   8,9,10,11




- To find the redundant bit R8, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R8 is an even number the value of R8(parity bit's value)=0. Thus, the data transferred is:




**Error detection and correction:** Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:




For all the parity bits we will check the number of 1's in their respective bit positions.

For R1: bits 1, 3, 5, 7, 9, 11. We can see that the number of 1's in these bit positions are 4 and that's even so we get a 0 for this.

For R2: bits 2,3,6,7,10,11 . We can see that the number of 1's in these bit positions are 5 and that's odd so we get a 1 for this.

For R4: bits 4, 5, 6, 7 . We can see that the number of 1's in these bit positions are 3 and that's odd so we get a 1 for this.

For R8: bit 8,9,10,11 . We can see that the number of 1's in these bit positions are 2 and that's even so we get a 0 for this.

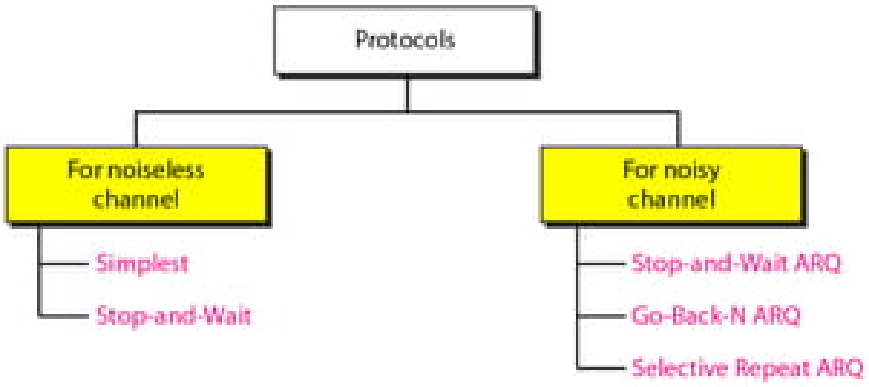
The bits give the binary number 0110 whose decimal representation is 6. Thus, bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.

**Flow Control**

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

**Error Control**

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.




## Noiseless Channels

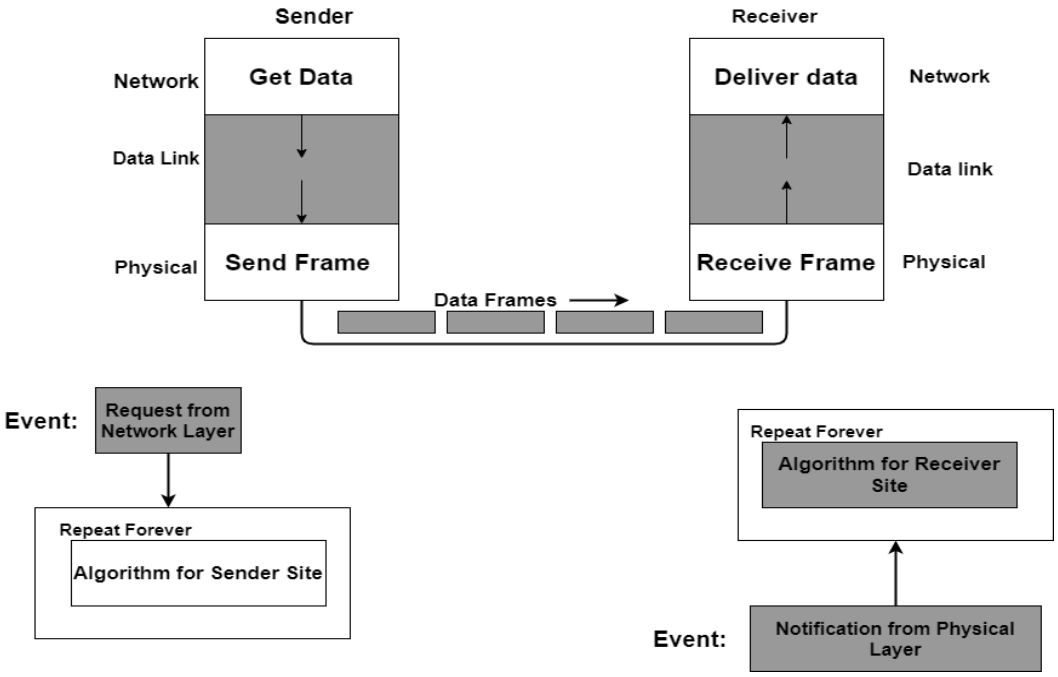
Noiseless Channel is an ideal channel in which no frames are lost, duplicated, or corrupted.There are two protocols for this type of channel:

1. Simplest Protocol
2. Stop and Wait Protocol

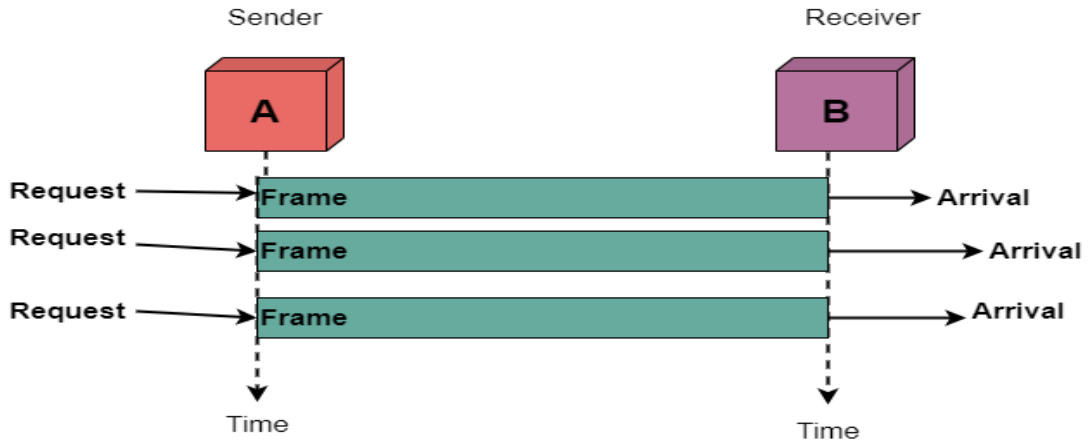
**Note**: In real world there are no Noiseless channels.

## Simplex (Simplest) Protocol:

1. Simplest Protocol has no flow control and no error control mechanism.
2. It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.
3. The data link layer at the **Sender site** gets data from its network layer, makes a frame outof the data, and sends it.
4. The data link layer at the **Receiver site** receives a frame from its physical layer, extractsdata from the frame, and delivers the data to its network layer.
5. The data link layers of the sender and receiver provide transmission services for theirnetwork layers.



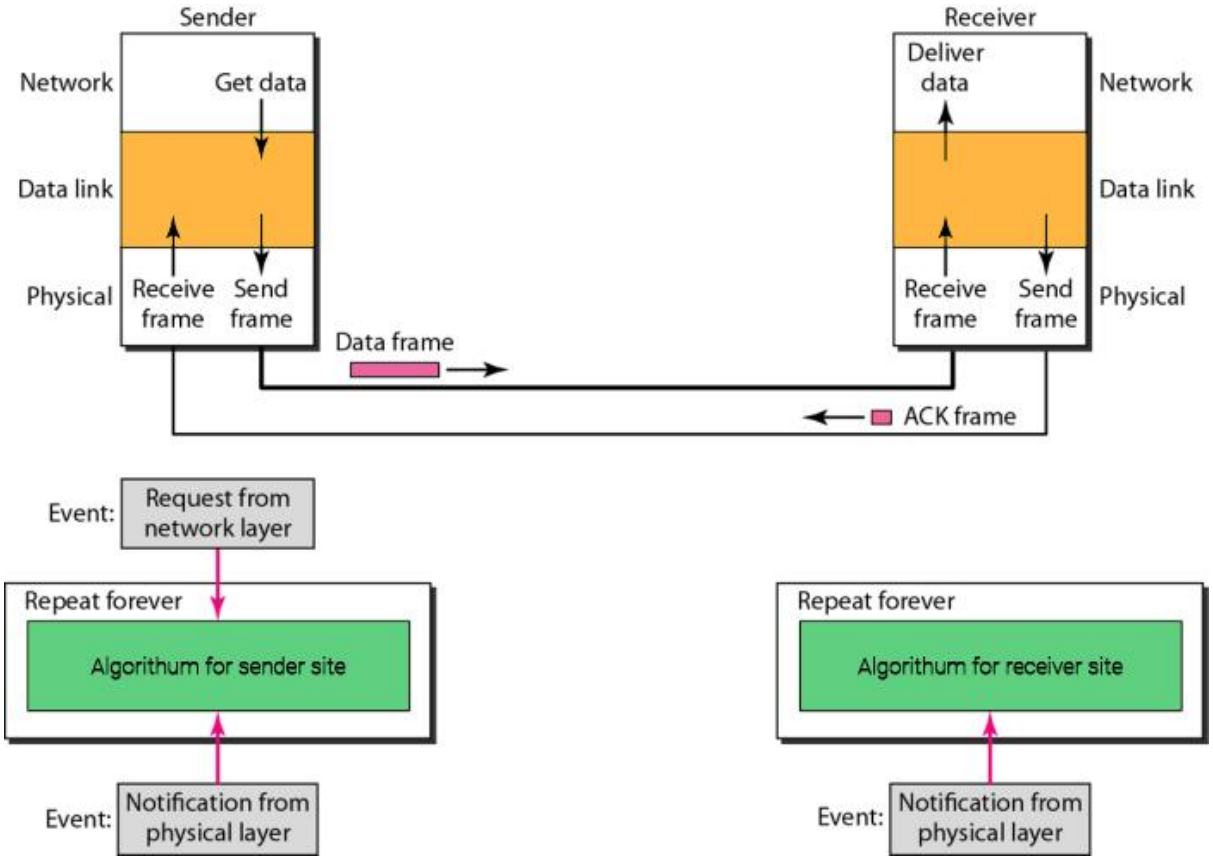

Using the simplest protocol the sender A sends a sequence of frames without even thinking about receiver .




## 2. Stop and Wait Protocol:

In Stop-and-Wait Protocol the sender sends one frame, stops until it receives confirmation (ACK's) from the receiver and then sends the next frame.

At any time, there is either one data frame on the forward channel or one ACK frame on thereverse channel. Therefore need a half-duplex link.




The sender sends one frame and waits for feedback from the receiver. When the ACK arrives,the sender sends the next frame.

**The algorithm used at the sender site for the stop-and-wait protocol**

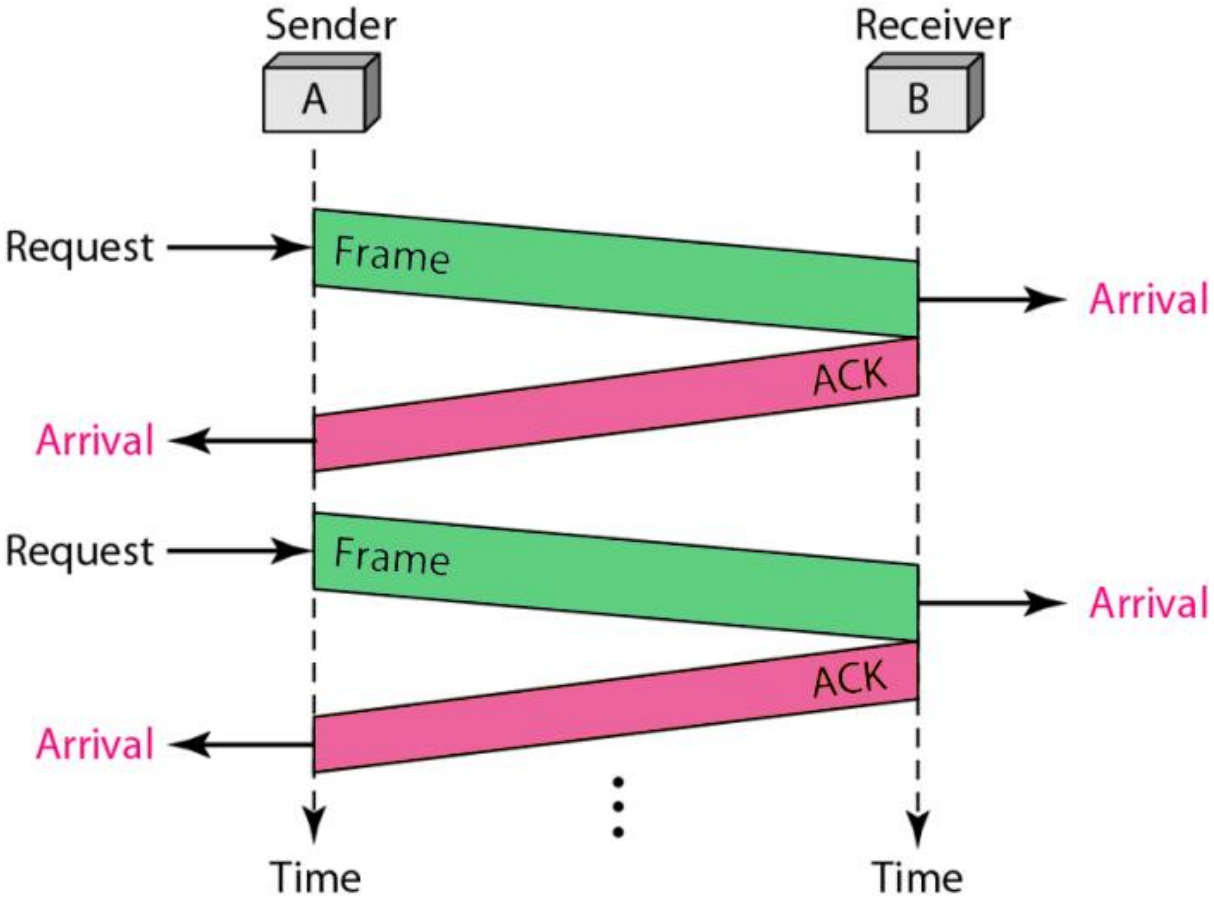
```
while(true)              //Repeat forever
canSend=true             //It will allow the first frame to go.
{
  WaitForEvent();        //sleep until the occurrence of an event
  if(Event(RequestToSend) AND canSend)
   {
     GetData();
     MakeFrame();
     SendFrame();   //Send the data frame
     canSend=false;   //cannot send until the acknowledgement arrives.
```

```
}
WaitForEvent();   //sleep until the occurrence of an event
if(Event(ArrivalNotification))  //indicates the arrival of the acknowledgement
  {
    ReceiveFrame();     //Means the ACK frame received
    canSend=true;
  }
     }
```

Algorithm At the Receiver Side

```
while(true)   //means Repeat forever
{
  WaitForEvent();  //sleep until the occurrence of an event
  if(Event(ArrivalNotification)) //indicates arrival of the data frame
  {
     ReceiveFrame();
     ExtractData();
     Deliver(data); //delivers the data to the network layer.
     SendFrame(); //Send the ACK frame
  }
     }
```

**Flow diagram of the stop-and-wait protocol**




**Disadvantages of Stop and Wait Protocol**

1. If the receiver does not respond when there is an error then sender doesn't know whichframe has to be resend.
2. In Stop and Wait protocol the sender does not send next frame until it receives ACK from receiver. If the ACK has lost then sender does not know when to send the frame.

## NOISY CHANNELS

In Noisy Channels we need to implement both Flow control and Error Control Mechanismsin the Protocols.

## Stop-and-Wait Automatic Repeat Request (ARQ)

This Protocol uses Acknowledgements and Sequence Numbers to implement Flow and ErrorControl mechanisms. The corrupted and lost frames need to be resent in this protocol.

There are 2 Questions arises:

1. **If the receiver does not respond when there is an error, how the sender knows whichframe to resend?**

***Solution:*** The sender keeps a copy of the sent frame. At the same time, it starts a timer.

If the timer expires and there is no ACK for the sent frame then **"3 actions"** to be performed

i. The frame is resent
ii. The copy is held
iii. The timer is restarted.

Note: Since the protocol uses the stop-and-wait mechanism, there is only one specific framethat needs an ACK even though several copies of the same frame can be in the network.

2. **What if an ACK frame is also lost?**

*Solution:*

Since an ACK frame can also be corrupted and lost, it needs a **Sequence Number** andRedundancy Bits.

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-orderone.

### Sequence Numbers

- Sequence Number is the number that is given to the data frame.
- A field is added to the data frame to hold the sequence number of that frame.
- The sequence numbers are based on modulo-2 arithmetic.

**Range:** Consider the field is m bits long, the range of sequence numbers are from 0 to $2^m$-1,and then the same number are repeated.

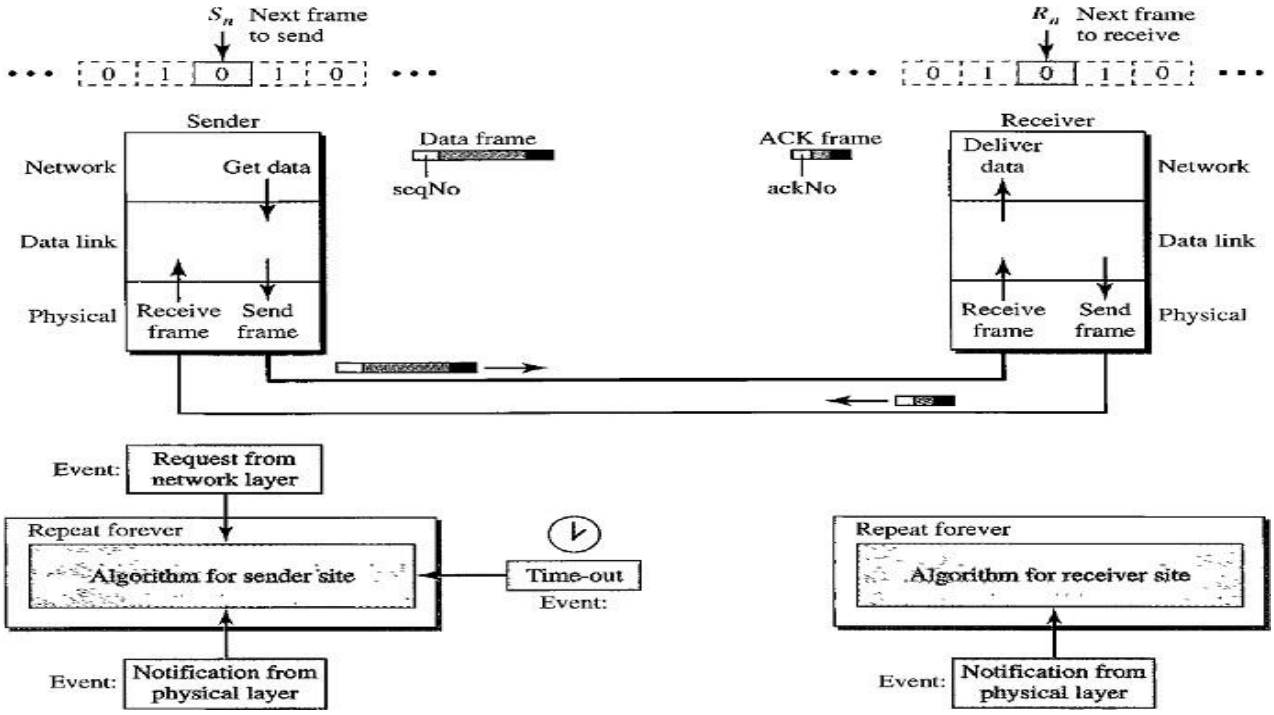
### Acknowledgment Numbers

The acknowledgment numbers always announce the sequence number of the next frameexpected by the receiver.

Example:

1. If frame 0 has arrived safe and sound, the receiver sends an ACK frame withacknowledgment 1 (meaning frame 1 is expected next).
2. If frame 1 has arrived safe and sound, the receiver sends an ACK frame withacknowledgment 0 (meaning frame 0 is expected).

**Design**




**Procedure:**

- The sending device keeps a copy of the last frame transmitted until it receives anacknowledgment for that frame.
- The design contains:
  - **seqNo**: Sequecne Number
  - **ackNo**: Acknowledgement Number
  - **$S_n$**: Sender Control variable- that holds the sequence number for the next frame to besent (0 or 1).
  - **$R_n$**: Receiver Control Variable- that holds the number of the next frame expected.
- When a frame is sent, the value of **$S_n$** is incremented (modulo-2), which means if it is 0, itbecomes 1 and vice versa.
- When a frame is received, the value of **$R_n$** is incremented (modulo-2), which means if it is0, it becomes 1 and vice versa.
- **$S_n$** variable points to the slot that matches the sequence number of the frame that has beensent, but not acknowledged.
- Sequence numbers are modulo-2, the possible numbers are 0,1,0,1,0,1,……and so on.
- **$R_n$** points to the slot that matches the sequence number of the expected frame.
- Three events can happen at the sender site; one event can happen at the receiver site.

**There are 3 Events can happen at Sender site.**

1. RequestToSend (which comes from network layer to data link layer)
2. ArrivalNotification (which comes from physical layer to data link layer)
3. TimeOut

**Event 1: Request to send**

- Event RequestToSend is used to send the frame to receiver.
- Before the frame is sent, it is stored in the buffer. We need at least one buffer to hold thisframe until sender make sure that it is received safe and sound.
- The copy is used for resending a corrupt or lost frame.
- Sender has to prevent the network layer from making a request before the previous frameis received safe and sound.

**Event 2: Arrival Notification**

- If the frame is not corrupted then three actions will be done:
  1. The ackNo of the ACK frame matches the sequence number of the next frame to send.
  2. Timer is stopped
  3. Purge the copy of the data frame that was saved.
- If the frame is corrupted then it ignore arrival notification event and wait for the nextevent to happen.

**Event 3: Time Out**

- After each frame is sent, a timer is started.
- When the timer expires the frame is resent and the timer is restarted.

**Receiver-site:**

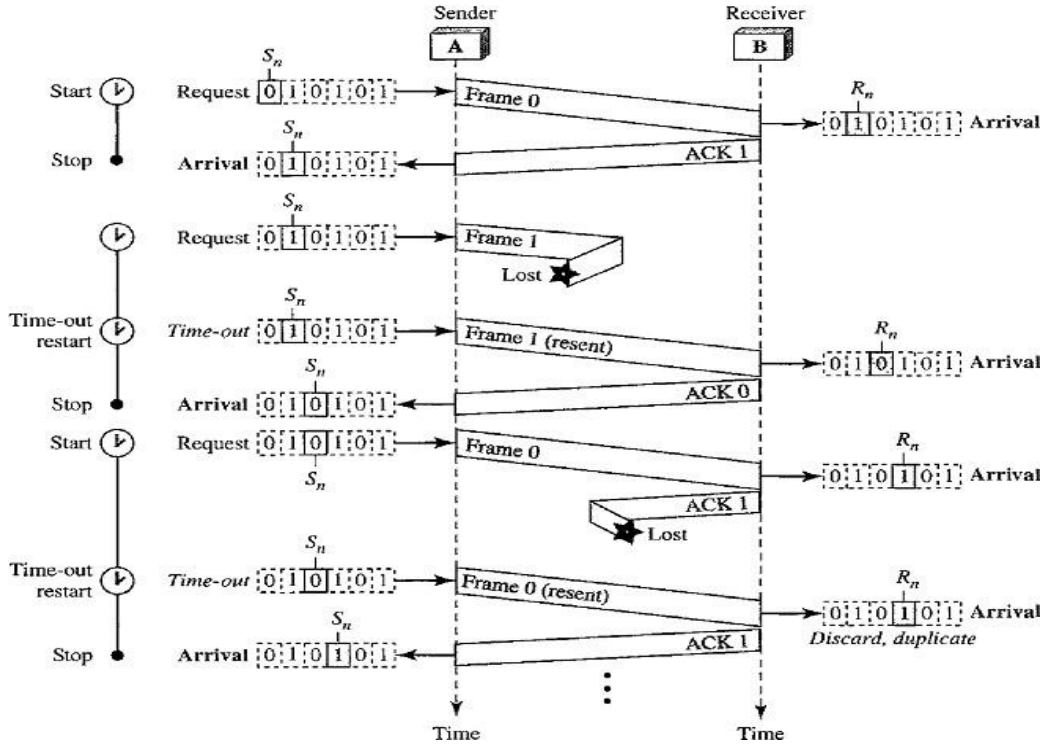
Only **One Event** happens at the receiver site.

- First, all arrived data frames that are corrupted are ignored.
- If the seqNo of the frame is the one that is expected ($\mathbf{R_n}$), the frame is accepted, the dataare delivered to the network layer, and the value of $\mathbf{R_n}$ is incremented.

**Note:**

- Even if the sequence number of the data frame does not match the next frame expected,an ACK is sent to the sender.
- This ACK reconfirms the previous ACK instead of confirming the frame received.
- This is done because the receiver assumes that the previous ACK might have been lost;the receiver is sending a duplicate frame.
- The resent ACK may solve the problem before the time-out does it.

**Flow diagram**




- Frame 0 is sent and acknowledged.
- Frame 1 is lost and resent after the time-out.
- The resent frame 1 is acknowledged and the timer stops.
- Frame 0 is sent and acknowledged, but the acknowledgment is lost.
- The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, itresends frame 0, which is acknowledged.

## Go-Back-N Automatic Repeat Request (ARQ)

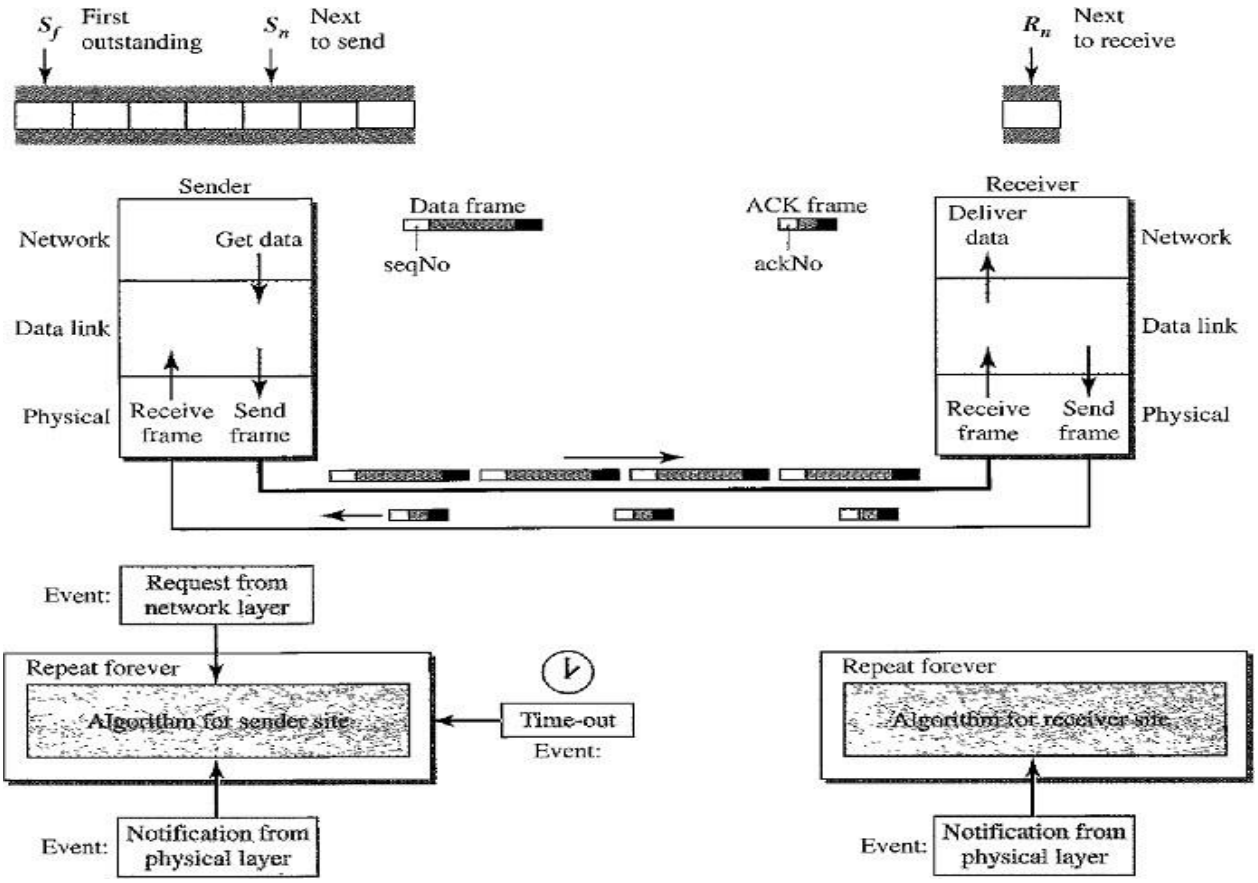
In this protocol

- Sender can send several frames before receiving acknowledgments.
- Sender keep a copy of these frames until the acknowledgments arrive.

## Design of Go-Back-N ARQ

The idea is similar to Stop-and-Wait ARQ, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction.

The difference is that the send window allows us to have as many frames in transition as there are slots in the send window.




This protocol uses the following concepts:

1. Sequence Numbers
2. Sliding Window
3. Timers
4. Acknowledgement
5. Retransmission (Resending a Frame)

## Sequence Numbers

- It is a number given to each frame included in the header.
- The header of the frame allows *m* bits for the sequence number, the sequence numbers range from 0 to $2^m$-1.
- The sequence numbers are repeated after the number $2^m$-1 (i.e) the sequence numbers are modulo-$2^m$

Example:
If m=4 , the sequence numbers are 0,1,2,3,4,5……14,15,0,1,2,3,4,5,6,……..

**Sliding Window**

Sliding Window is an abstract concept that defines the sender and receiver needs to deal with only part of the possible sequence numbers.

There are 2 types of windows are used:

i. Send Window
ii. Receive Window

The range that is the concern of sender is called the send sliding window or Send Window

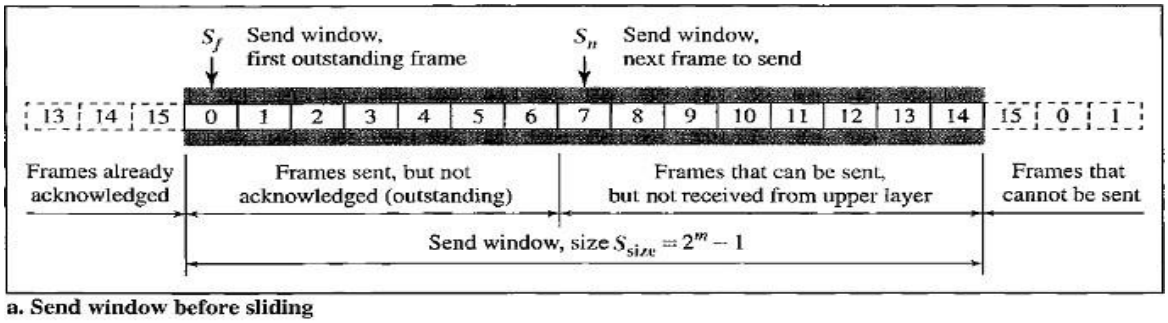
The range that is the concern of receiver is called receive sliding window or receive window.

**Send Window**

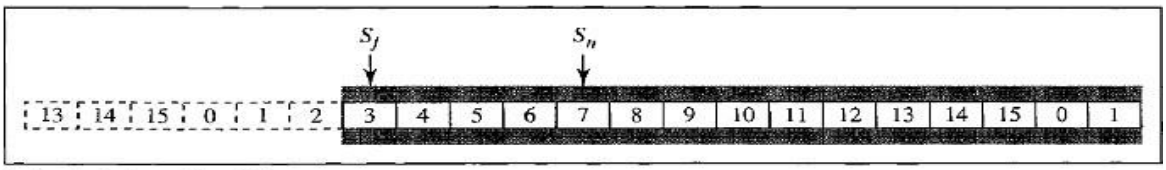
- The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit.
- In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent.
- The maximum size of the send window is $\mathbf{2^m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

Example: Let us take m=4. Size of window=15.

Consider the below figure:




**a. Send window before sliding**




**b. Send window after sliding**

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

1. The first region (the left side of the window) defines the sequence numbers belonging to frames that are already acknowledged. The sender don"t need to keep copies of the frames.
2. The second region defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. These frames are called outstanding frames.
3. The third range defines the range of sequence numbers for frames that can be sent. The corresponding data packets have not yet been received from the network layer.
4. The fourth region defines sequence numbers that cannot be used until the window slides.

The window uses three variables define its size and location at any time.

i. $S_f$ defines the sequence number of the first (oldest) outstanding frame.
ii. $S_n$ holds the sequence number that will be assigned to the next frame to be sent.
iii. $S_{size}$ defines the size of the window, which is fixed in our protocol.

The acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame.

In the above figure frames 0, 1, and 2 are acknowledged, so the window has slid to the right three slots.

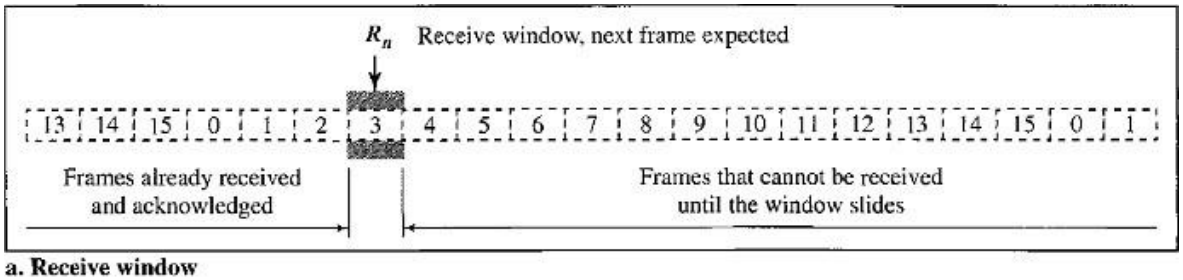
**Receive Window**

- The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent.
- The size of the receive window is always 1.
- The receiver is always looking for the arrival of a specific frame.
- Any frame arriving out of order is discarded and needs to be resent.

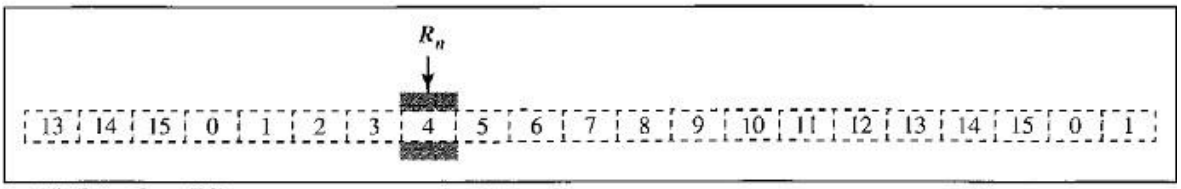
There is only one variable required, that is $R_n$

$R_n$: defines next frame expected, belongs to Receive Window.

- The sequence numbers to the left of the window belong to the frames already received and acknowledged.
- The sequence numbers to the right of this window define the frames that cannot be received.
- Any received frame with a sequence number in these two regions is discarded.
- Only a frame with a sequence number matching the value of $R_n$ is accepted and acknowledged. (i.e. $S_n=R_n$)The receive window slides only one slot at a time, when the correct frame is received thewindow slides.



a. Receive window



b. Window after sliding

**Timers**

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 frame is received out of order, the receiver is silent and will discard

all subsequent frames until it receives the one it is expecting.

- The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This causes the sender to go back and resend all frames, beginning with the one with the expired timer.
- The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

**Retransmission (Resending a Frame)**

When the timer expires, the sender resends all outstanding frames.

Example:

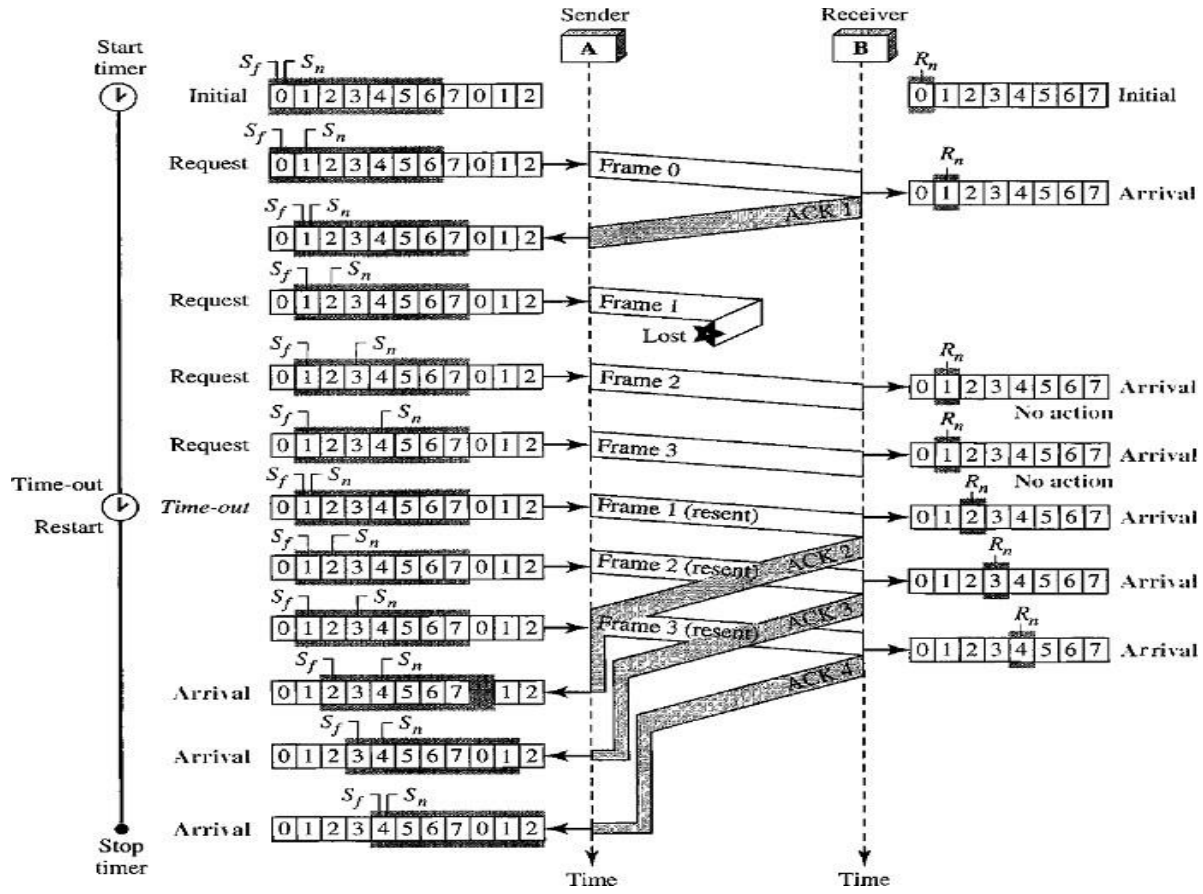
- Suppose the sender has already sent frame 6, but the timer for frame 3 expires.
- This means that frame 3 has not been acknowledged.
- The sender goes back and sends frames 3, 4,5, and 6 again.
- That is why the protocol is called ***Go-Back-N*** **ARQ**.

**Flow Diagram**

The below shows what happens when a frame is lost.

- Frames 0, 1, 2, and 3 are sent.
- Frame 0 is acknowledged but **Frame 1** is lost.
- The receiver receives frames 2 and 3, but they are discarded because they are received out of order (frame 1 is expected).
- The sender receives no acknowledgment about frames 1, 2, or 3.
- Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know whether the frame is lost or corrupted.
- Note that the resending of frames l, 2, and 3 is the response to one single event.
- When the sender is responding to this event, it cannot accept the triggering of other events.
- This means that when ACK 2 arrives, the sender is still busy with sending frame 3.
- The physical layer must wait until this event is completed and the data link layer goes back to its sleeping state.
- Vertical line in the figure is to indicate the delay.

Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.




## Disadvantage with Go-Back-N ARQ

*Go-Back-N* ARQ protocol is very inefficient for a noisy link.

- *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 and they are simply discarded.
- In a noisy link a frame has a higher probability of damage; so it leads to the resending of multiple frames.
- This resending uses more bandwidth and slows down the transmission. This is a major disadvantage.

**Solution:** When there is just one frame is damaged, then only the damaged frame is resent, instead of resending from **N$^{th}$** frame. This will be achieved by using **Selective Repeat ARQ** Protocol.

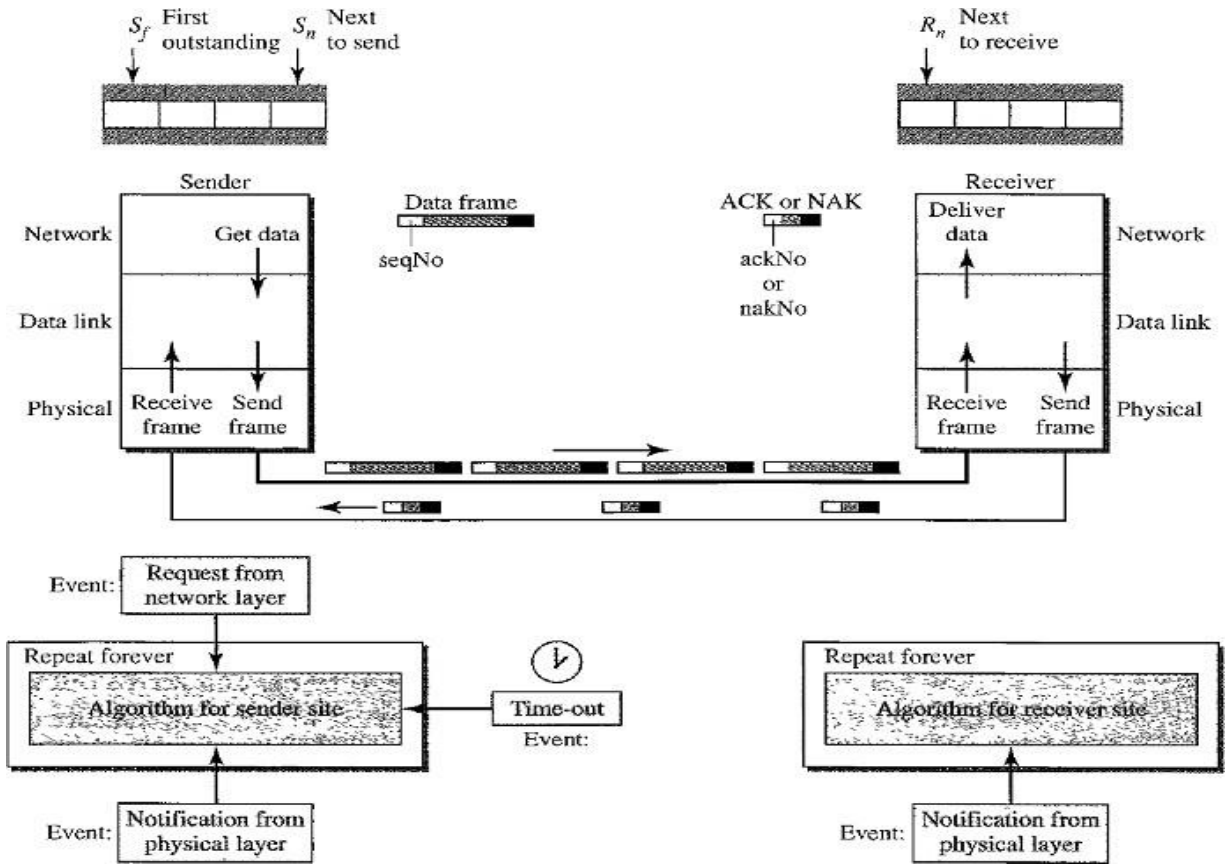
## SELECTIVE REPEAT Automatic Repeat Request

It is more efficient for Noisy links but processing at the receiver is more complex.

## Design

### Window Sizes

Window size **$2^{m-1}$** means the size of the sender and receiver window is at most one half of **$2^m$** **(i.e $2^m$/2)**.



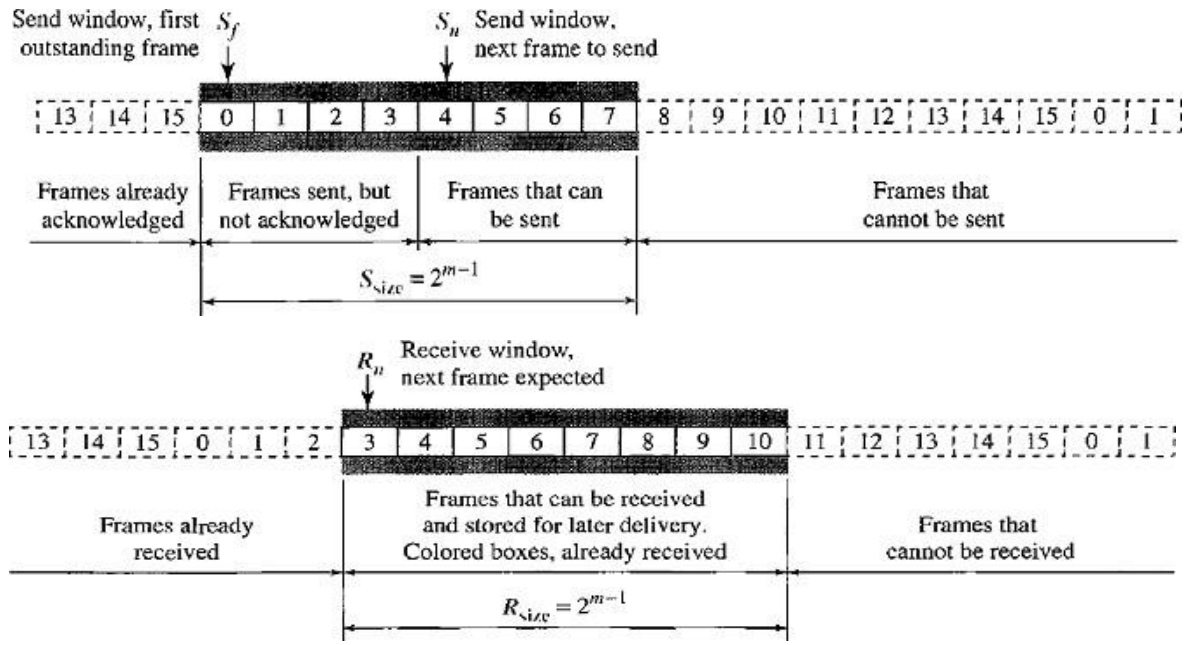

**Windows**

It uses two windows:

1. Send Window
2. Receive Window

The size of the send window and receive window is same as $2^{m-1}$.

**Example**: If $m = 4$, the sequence numbers ranges from 0 to 15, but the size of the window is just 8.



- The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer.
- Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered.
- We need to mention that the receiver never delivers packets out of order to the network layer.
- Those slots inside the receive window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

**Sender site:**

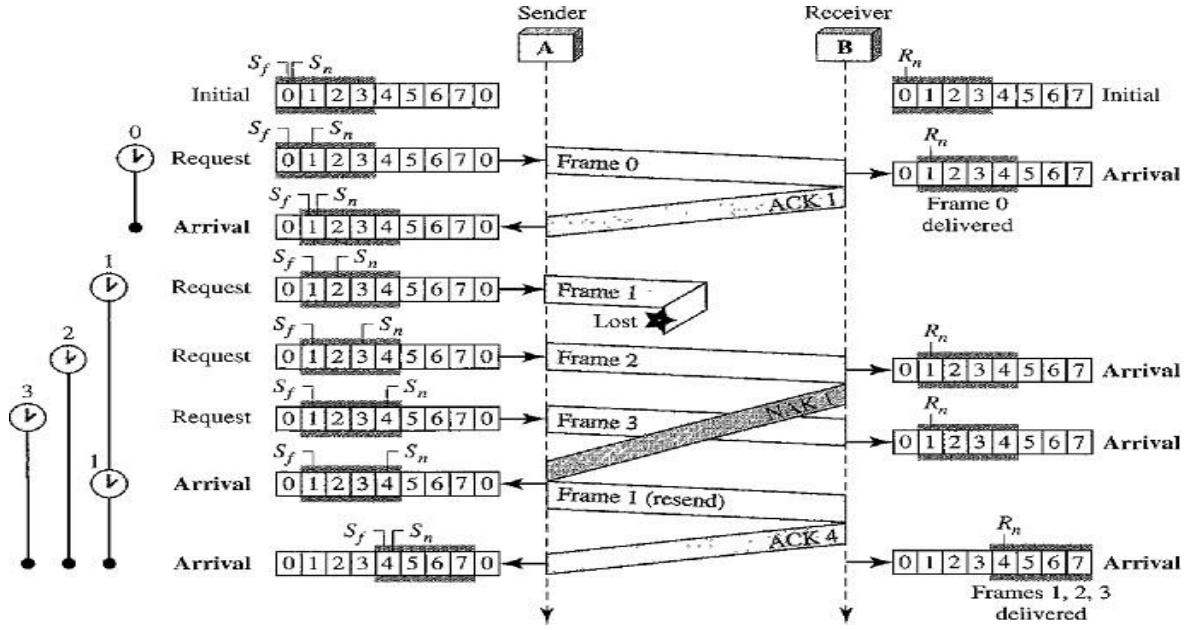
- For every request event a timer is started before sending any frame. The arrival event is more complicated. An ACK or a NAK frame may arrive at the sender site.
- If a valid NAK frame arrives, we just resend the corresponding frame.
- If a valid ACK arrives 3 actions will be done:
  1. Purge the buffers
  2. Stop the corresponding timer
  3. Move the left wall of the window.
- The time-out event is simpler, only the frame which times out is resent.

**Receiver site:**

- Receiver sends ACK and NAK frames to sender.
- If the Receiver receives a corrupted frame and a NAK has not yet been sent, Receiver sends a NAK to tell the sender that we have not received the frame we expected.
- If the frame is not corrupted and the sequence number is in the window, Receiver stores the frame and marks the slot.
- If contiguous frames, starting from $R_n$ have been marked, Data link layer delivers their data to the network layer and slide the window.

**Flow Diagram**

By looking at the below flow diagram, we can observe below points:



**Timers**

- Each frame sent or resent needs a timer and the timers needs to be numbered such as 0,1,2,3..etc.
- The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives.
- The timer for frame 1 starts at the second request and restarts when a NAK arrives, and finally stops when the last ACK arrives.
- The other two timers start when the corresponding frames are sent and stop at the last arrival event.

**Receiver Site**

- At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer.
- At the second arrival, frame 2 arrives and is stored and marked (colored slot), but it cannot be delivered because frame 1 is missing.
- At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered.
- Only at the last arrival, when finally a copy of frame 1 arrives and frames 1, 2, and 3 be delivered to the network layer.
- There are two conditions for the delivery of frames to the network layer:
  i. a set of consecutive frames must have arrived.
  ii. the set starts from the beginning of the window.
- After the first arrival, there was only one frame and it started from the beginning of the window.
- After the last arrival, there are three frames and the first one starts from the beginning of the window.

**Importance of NAK's**

- Here a NAK is sent after the second arrival, but not after the third.
- The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames.
- The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done.
- The first NAK sent is remembered and is not sent again until the frame slides.
- A NAK is sent once for each window position and defines the first slot in the window.

**Importance of ACK's**

- There are only two ACKs are sent here. The first one acknowledges only the first frame The second one acknowledges three frames.
- In Selective Repeat, ACKs are sent when data are delivered to the network layer.
- If the data belonging to *n* frames are delivered in one shot, only one ACK is sent for all of them.
- In the above figure frame 1,2,3, are sent to Network layer and then ACK4 is sent, to represent that Frame1, Frame2, Frame3 are delivered.

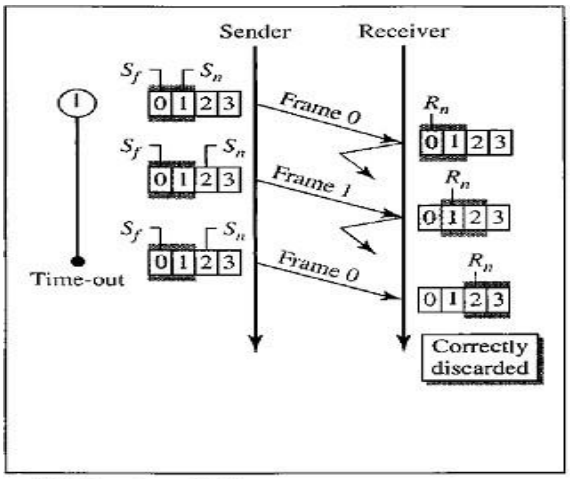
There is a question arises that :

**Why the size of the sender and receiver windows is $2^{m-1}$?**

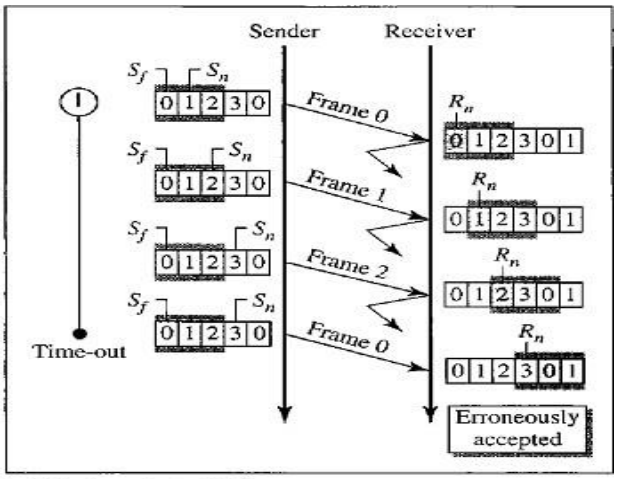
Sol: For an example, take $m$ = 2, which means the size of the window is **$2^m/2$**, or 2.Now we compare window size=2 and window size =3.

**Window size=2**

- If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0expires and frame 0 is resent.
- The window of the receiver is now expecting frame 2, not frame 0, so this duplicate frameis correctly discarded.




a. Window size = $2^{m-1}$




b. Window size > $2^{m-1}$

**Window Size=3**

- When the size of the window is 3 and all acknowledgments are lost, the sender sends aduplicate of frame 0.
- However, this time, the window of the receiver expects to receive frame 0 (0 is part of thewindow), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle.
- This is clearly an error.

**PIGGYBACKING**

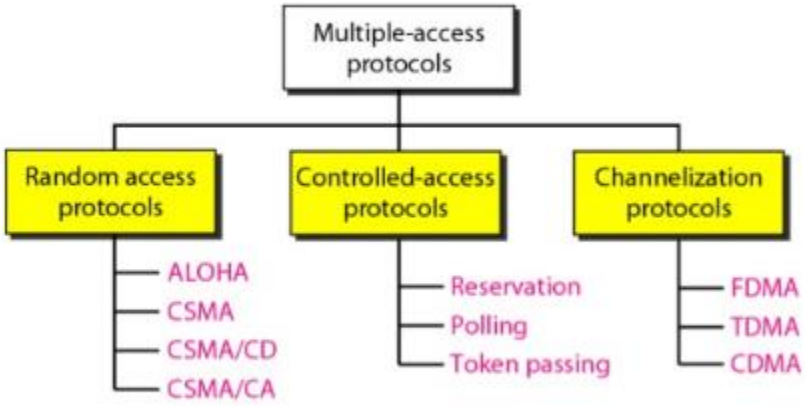
- The protocols Stop-and-wait ARQ, Go-Back-N ARQ, Selective Repeat ARQ are all unidirectional. That means data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction.
- In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. This technique called **piggybacking.**
- **Piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information from B to A. Similarly when a frame is carrying data from B to A, it can also carry control information frames from A to B.

The Data Link Layer can be divided into 2 sublayers:

1. Data Link Control
2. Multiple Access Resolution

- The Multiple access resolution layers is responsible for resolving access to the shared media.

These protocols are categorized as below:




## Random Access Protocols

Random Access method is also called Contention method.

- There is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called **Random Access methods**.
- There are no rules to specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called **Contention Methods**.

If more than one station tries to send, there is an **Access Conflict-Collision** and the frames will be either destroyed or modified during the collision.

In order to avoid these collisions 4 protocols are implemented, they are:

- ALOHA
- CSMA
- CSMA/CD
- CSMA/CA

## ALOHA

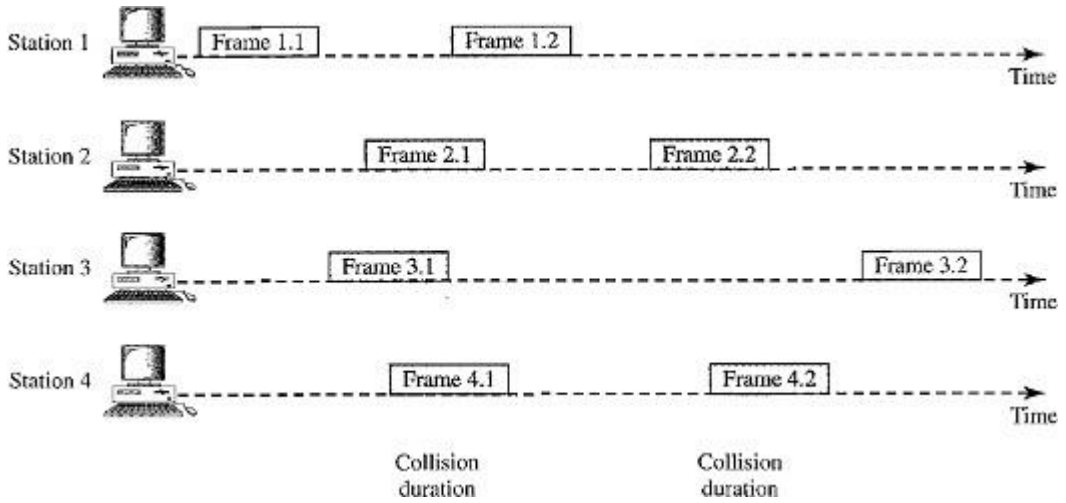
ALOHA was developed at the University of Hawaii in early **1970**. It was designed for a Wireless radio LAN, but it can be used on any shared medium. Due to shared medium there are potential collisions in this arrangement.

There are two types of methods in ALOHA 1. Pure ALOHA 2. Slotted ALOHA

## Pure ALOHA

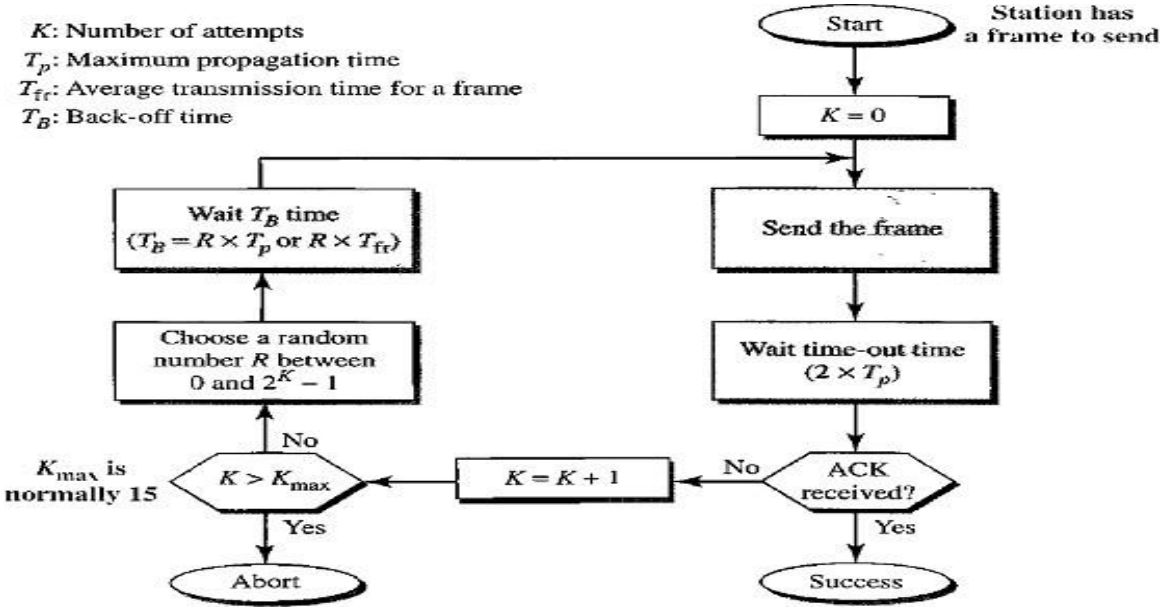
The original ALOHA or Pure ALOHA is a simple protocol.

The idea is that each station sends a frame whenever it has a frame to send, there is only one channel to share, and there is the possibility of collision between frames from different stations.




- In the above figure, there are four stations each sending two frames and shares the same channel. Some of these frames collide because multiple frames are in contention for the shared channel.
- By observing the above figure only 2 frames frame 1.1 and frame 3.2 can be delivered at receiver, and the remaining frames collide with each other and they are lost or discarded at the receiver side.
- The pure ALOHA protocol relies on Acknowledgments (ACK) 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; this time is called the Back-Off Time $\mathbf{T_B}$.
- Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts $\mathbf{K_{max}}$ a station must give up and try later.

The procedure for pure ALOHA is given in the figure:




- The time-out period is equal to the maximum possible round-trip propagation delay,which is twice the amount of time required to send a frame between the two most widely separated stations (2 x $T_p$).
- The back-off time $T_B$ is a random value that normally depends on $K$ (the number of attempted unsuccessful transmissions).
- For each retransmission, a multiplier in the range 0 to $2^K - 1$ is randomly chosen and multiplied by $T_p$ (maximum propagation time) or $T_{fr}$ (Frame transmission time or the average time required to send out a frame) to find $T_B$.
- Note that in this procedure, the range of the random numbers increases after each collision. The value of $K_{max}$ is usually chosen as **15**.

**Vulnerable time**

Vulnerable time is the length of time, in which there is a possibility of collision.

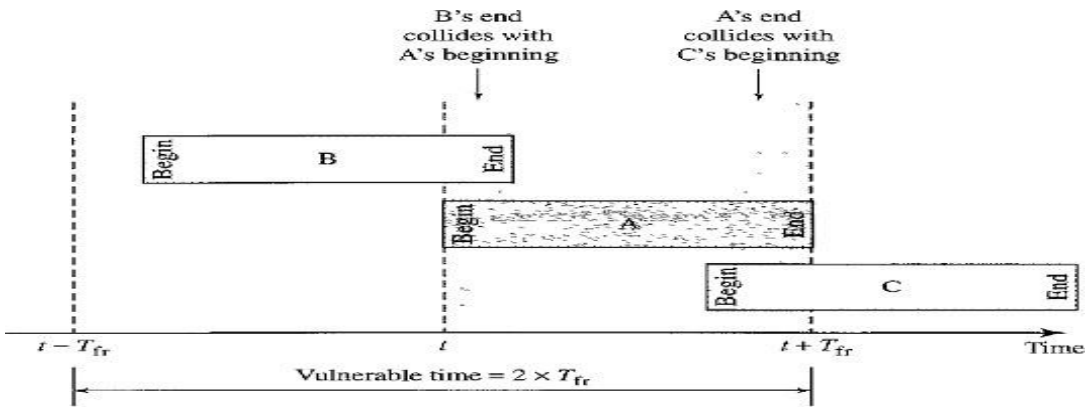
Let us assume that the stations send fixed-length frames with each frame taking $T_{fr}$**'s** to send.

Station A sends a frame at time **t.**

Now station B has already sent a frame between **t** - $T_{fr}$ and **t**.

This leads to a collision between the frames from station A and station B.

The end of B's frame collides with the beginning of A's frame.




Suppose that station C sends a frame between **t** and **t + $T_{fr}$** .

Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

The vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

| Pure ALOHA vulnerable time = **2 x $T_{fr}$** |
|---|

**Throughput**

- The throughput for pure ALOHA is **$S = G \times e^{-2G}$**.
- The maximum throughput **$S_{max}$** = 0.184 when G = (½).
- (i.e.) one frame is generated during two frame transmission times, then 18.4 percent of these frames reach their destination successfully.

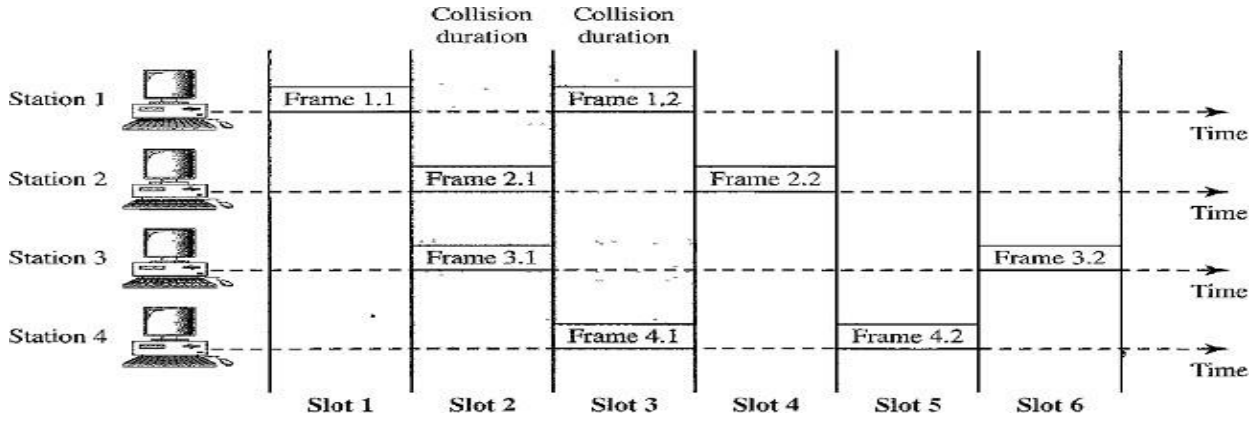
## Slotted ALOHA

Pure ALOHA has a vulnerable time of **2 x $T_{fr}$** . This is so because there is no rule that defines when the station can send.

- A station may send soon after another station has started or soon before another station has finished.
- Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
- In slotted ALOHA we divide the time into slots of **$T_{fr}$**"s and force the station to send only at the beginning of the time slot.
- 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.
- There is still the possibility of collision if two stations try to send at the beginning of the same time slot.

**i.e.** the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

Slotted ALOHA vulnerable time = **$T_{fr}$.**




Below figure shows an example of frame collisions in slotted ALOHA.

### Throughput

- The throughput for slotted ALOHA is $S =: G \times e^{-G}$.
- The maximum throughput $S_{max} == 0.368$ when G =1.
- If a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully.

**Example**:

Consider a system generating 20 bit frames and connected through a shared 20kbps channel. Find throughput in percent if slotted ALOHA is used and frame rate is 1000 fps.

Frame size = L = 20 bits
Rate = R = 20kbps
Transmission time,T = L/R = $1 * 10^{-3}$ s
Throughput, S = G $e^{-G}$, where G= Number of frames per T
So, G = $1000 * 10^{-3} = 1$
Therefore, S = $e^{-1}$ = 0.368 = 36.8%

## CSMA (Carrier Sense Multiple Access)

It is a **carrier sense multiple access** based on media access protocol to sense the traffic on a channel (idle or busy) before transmitting the data. It means that if the channel is idle, the station can send data to the channel. Otherwise, it must wait until the channel becomes idle. Hence, it reduces the chances of a collision on a transmission medium.

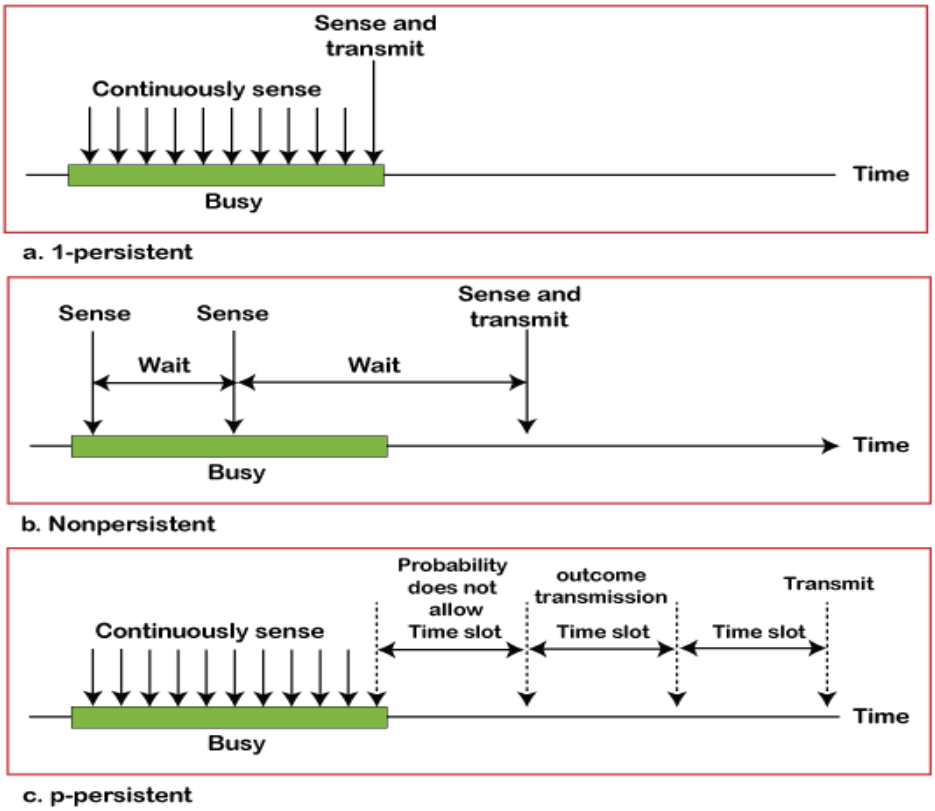
### CSMA Access Modes

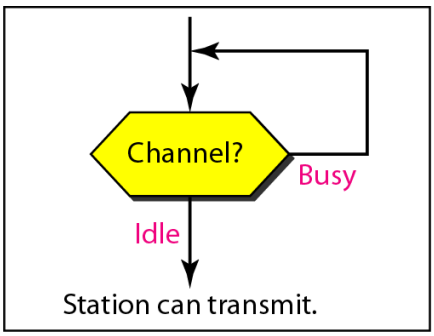
**1-Persistent:** In the 1-Persistent mode of CSMA that defines each node, first sense the shared channel and if the channel is idle, it immediately sends the data. Else it must wait and keep track of the status of the channel to be idle and broadcast the frame unconditionally as soon as the channel is idle.

**Non-Persistent:** It is the access mode of CSMA that defines before transmitting the data, each node must sense the channel, and if the channel is inactive, it immediately sends the data. Otherwise, the station must wait for a random time (not continuously), and when the channel is found to be idle, it transmits the frames.

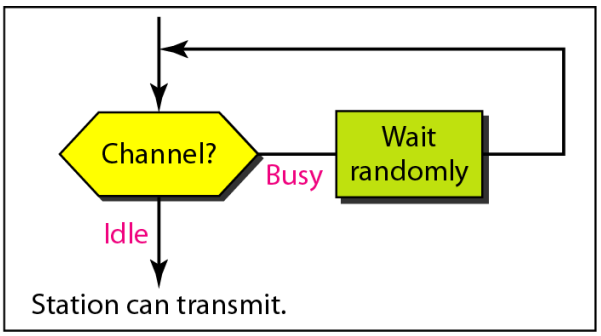
**P-Persistent:** It is the combination of 1-Persistent and Non-persistent modes. The P-Persistent mode defines that each node senses the channel, and if the channel is inactive, it sends a frame with a **P** probability. If the data is not transmitted, it waits for a (**q = 1-p probability**) random time and resumes the frame with the next time slot.

**O- Persistent:** It is an O-persistent method that defines the superiority of the station before the transmission of the frame on the shared channel. If it is found that the channel is inactive, each station waits for its turn to retransmit the data.

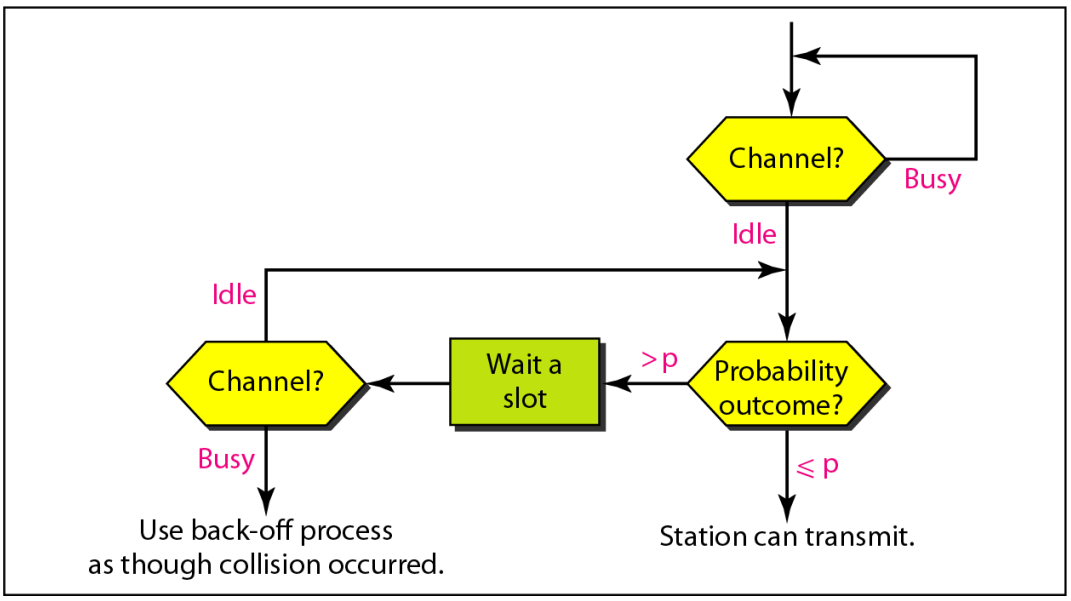





a. 1-persistent




b. Nonpersistent




c. p-persistent

**CSMA/ CD**

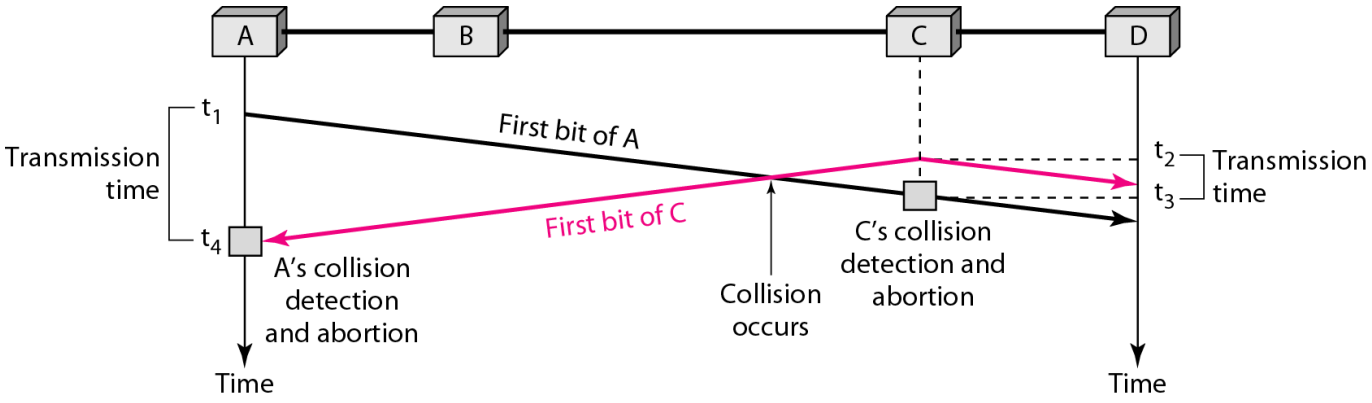
It is a **carrier sense multiple access/ collision detection** network protocol to transmit data frames. The CSMA/CD protocol works with a medium access control layer. Therefore, it first senses the shared channel before broadcasting the frames, and if the channel is idle, it transmits a frame to check whether the transmission was successful. If the frame is successfully received, the station sends another frame. If any collision is detected in the CSMA/CD, the station sends a jam/ stop signal to the shared channel to terminate data transmission. After that, it waits for a random time before sending a frame to a channel.

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

CSMA method does not specify the procedure following a collision but 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 it is successful the station is finished.
- If it not successful and there is a collision, the frame is sent again.




- At time **t 1**, station A has executed its persistence procedure and starts sending the bits of its frame.
- At time **t2**, station C has not yet sensed the first bit sent by A.
- Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right.
- The collision occurs sometime after time **t2** Station C detects a collision at time **t3** when it receives the first bit of A's frame. Station C immediately aborts transmission.
- Station A detects collision at time **t4** when it receives the first bit of C's frame; it also immediately aborts transmission.
- A transmits for the duration **t4 – t1**; C transmits for the duration **t3 - t2**
- At time **t4,** the transmission of A‟s frame is aborted; at time t3, the transmission of C's frame is aborted. Both are incomplete.
- This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection.

## Minimum Frame Size (2Tp)

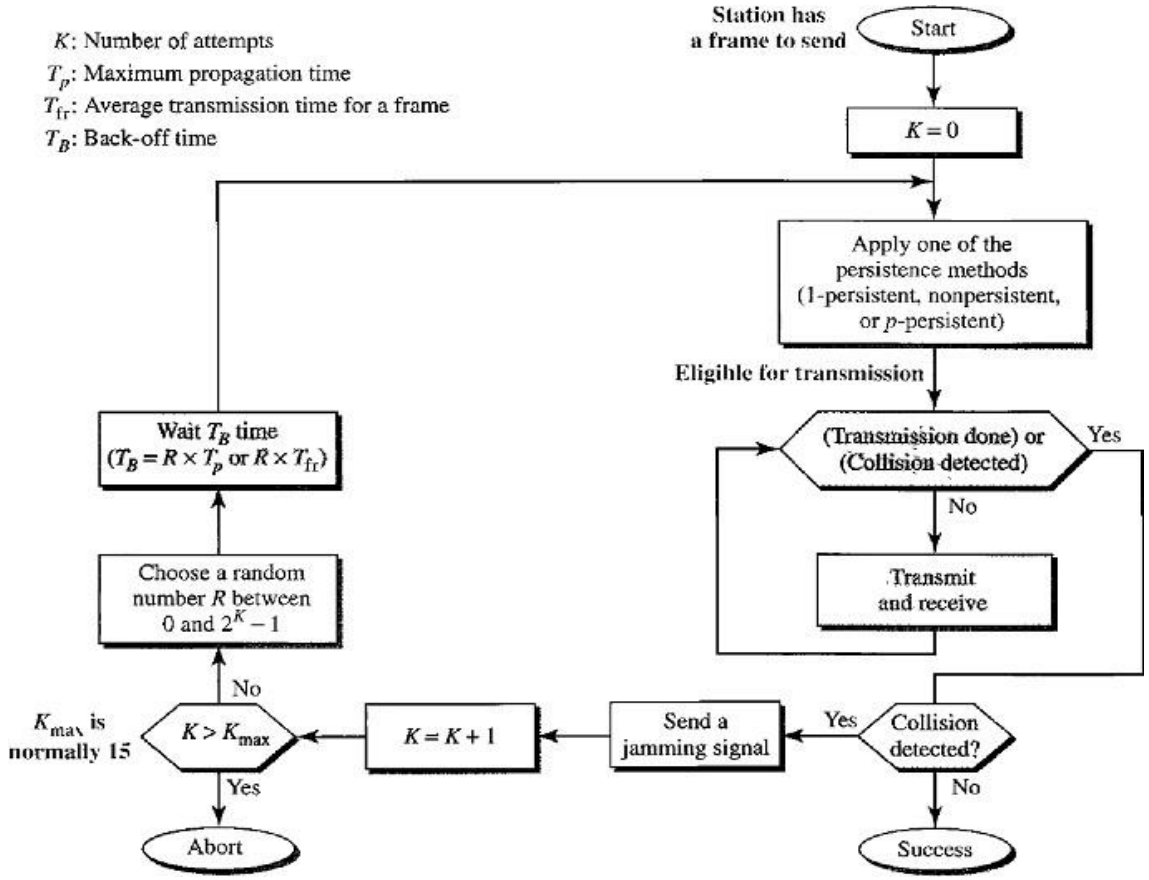
If the two stations involved in a collision are the maximum distance apart, the signal from the first station takes Tp time to reach the second station and the effect of the collision takes another Tp time to reach the first station.

Therefore, the frame transmission time $\mathbf{T_{fr}}$ must be at least two times the maximum propagation time Tp. So the first station must still be transmitting after 2Tp .

**Procedure**

- We need to sense the channel before we start sending the frame by using one of the persistence processes.
- In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In *CSMA/CD*, transmission and collision detection is a continuous process.
- We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously using two different ports.
- We use a loop to show that transmission is a continuous process.
- We constantly monitor in order to detect one of two conditions:
  either transmission is finished or a collision is detected. Either event stops transmission.
- When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.

- Here we send a short jamming signal that enforces the collision in case other stations have not yet sensed the collision.



## CSMA/ CA

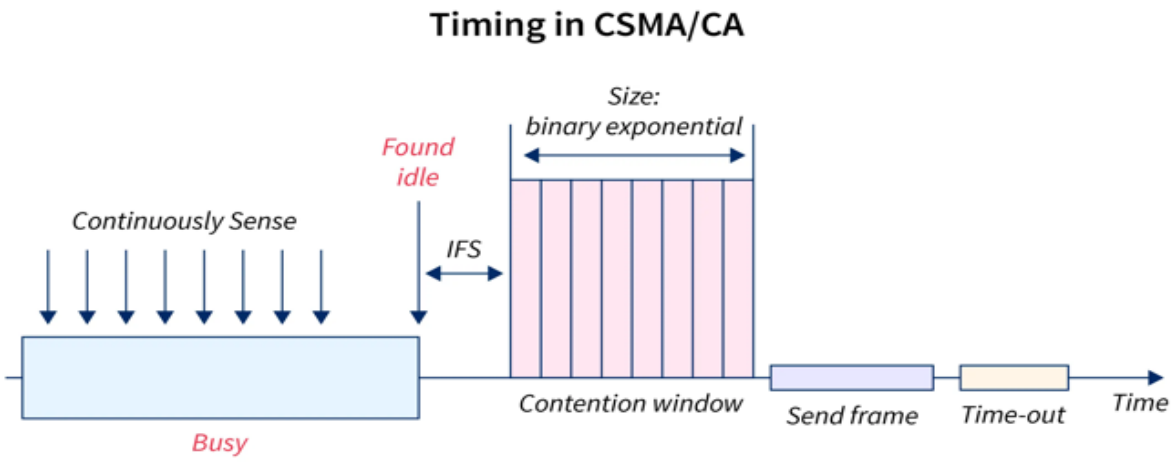
It is a **carrier sense multiple access/collision avoidance** network protocol for carrier transmission of data frames. It is a protocol that works with a medium access control layer. When a data frame is sent to a channel, it receives an acknowledgment to check whether the channel is clear. If the station receives only a single (own) acknowledgments, that means the data frame has been successfully transmitted to the receiver. But if it gets two signals (its own and one more in which the collision of frames), a collision of the frame occurs in the shared channel. Detects the collision of the frame when a sender receives an acknowledgment signal.

Following are the methods used in the CSMA/ CA to avoid the collision:

**Interframe space**: In this method, the station waits for the channel to become idle, and if it gets the channel is idle, it does not immediately send the data. Instead of this, it waits for some time, and this time period is called the **Interframe** space or IFS. However, the IFS time is often used to define the priority of the station.

**Contention window**: In the Contention window, the total time is divided into different slots. When the station/ sender is ready to transmit the data frame, it chooses a random slot number of slots as **wait time**. If the channel is still busy, it does not restart the entire process, except that it restarts the timer only to send data packets when the channel is inactive.

**Acknowledgment**: In the acknowledgment method, the sender station sends the data frame to the shared channel if the acknowledgment is not received ahead of time.



## WIRED LANS: ETHERNET (802.3)

Local Area Network (LAN) is a computer network that is designed for limited geographic area such as building or campus. LAN is a shared resource.

LAN technologies: Ethernet, Token Ring, Token Bus, FDD, ATM LAN.

## IEEE STANDARDS for LAN

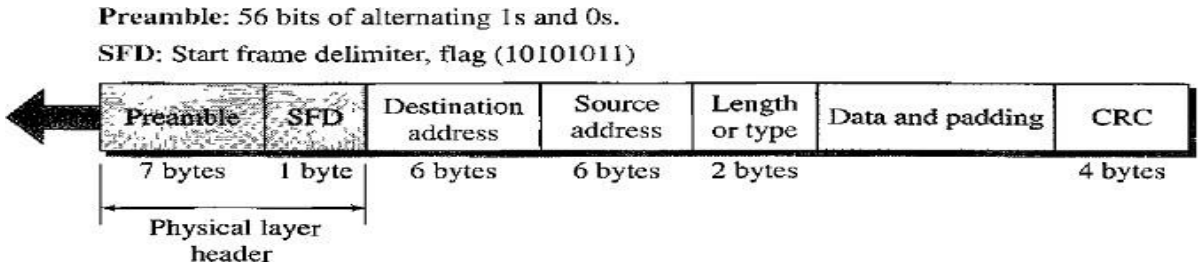
The IEEE has subdivided the data link layer into two sublayers:

1. Logical Link Control(LLC)
2. Media Access Control(MAC)

**Note:** A frame defined in HDLC is divided into a PDU at the LLC sub layer and a frame at the MAC sub-layer

### Frame Format

The Ethernet frame contains **seven** fields: Preamble, SFD, DA, SA, Length or Type of protocol data unit (PDU), Upper-layer data, and CRC.




### Preamble

- The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0"s and 1"s that alerts the receiving system to the coming frame and enables it to synchronize its input timing.
- The pattern provides only an alert and a timing pulse.
- The 56-bit pattern allows the stations to miss some bits at the beginning of the frame.

Note: The preamble is actually added at the physical layer and is not (formally) part of the frame.

**Start Frame Delimiter (SFD)**

- The second field (l byte: 10101011) signals the beginning of the frame.
- The SFD warns the 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 (i.e.) 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. Both uses are common today.
- 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.

**Data**

- This field carries data encapsulated from the upper-layer protocols.
- It is a minimum of 46 bytes and a maximum of 1500 bytes.

**CRC**

- The last field contains Error Detection information. The Length of this field is 4 byte (32-bit) hence a CRC-32 is used.

## Addressing

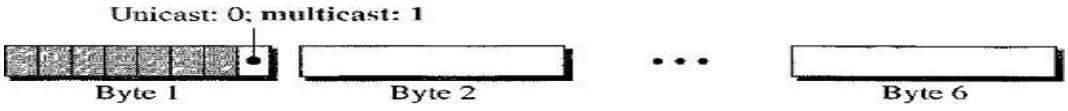
- Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card(NIC).
- The NIC fits inside the station and provides the station with a 6-byte physicaladdress.
- The Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between thebytes.

Example: **06 : 01 : 02 : 01 : 2C : 4B**

There are 3 types addressing:

1. UnicastAddressing
2. MulticastAddressing
3. BroadcastAddressing

- A source address is always a unicast address-the frame comes from only onestation.
- The destination address can be unicast, multicast, orbroadcast.
- If the least significant bit of the first byte in a destination address is 0, the addressis **Unicast Address,** otherwise (i.e. 1) it is **Multicast**.



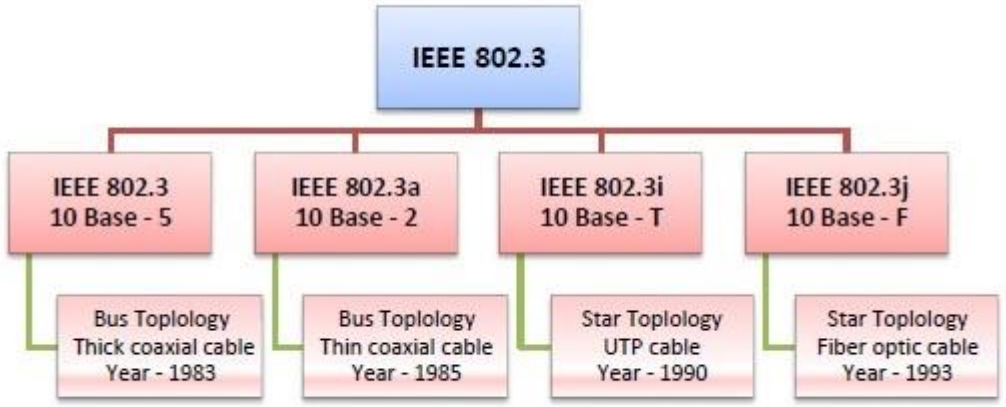
**Unicast Destination Address** defines only one recipient; the relationship between the sender and the receiver is one-to-one.

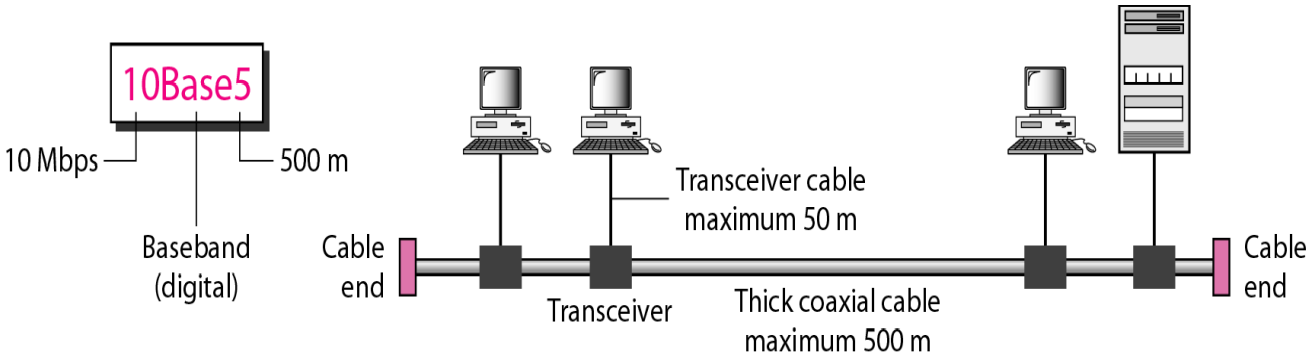
**Multicast Destination Address** defines a group of addresses; the relationship between the sender and the receivers is one-to-many.

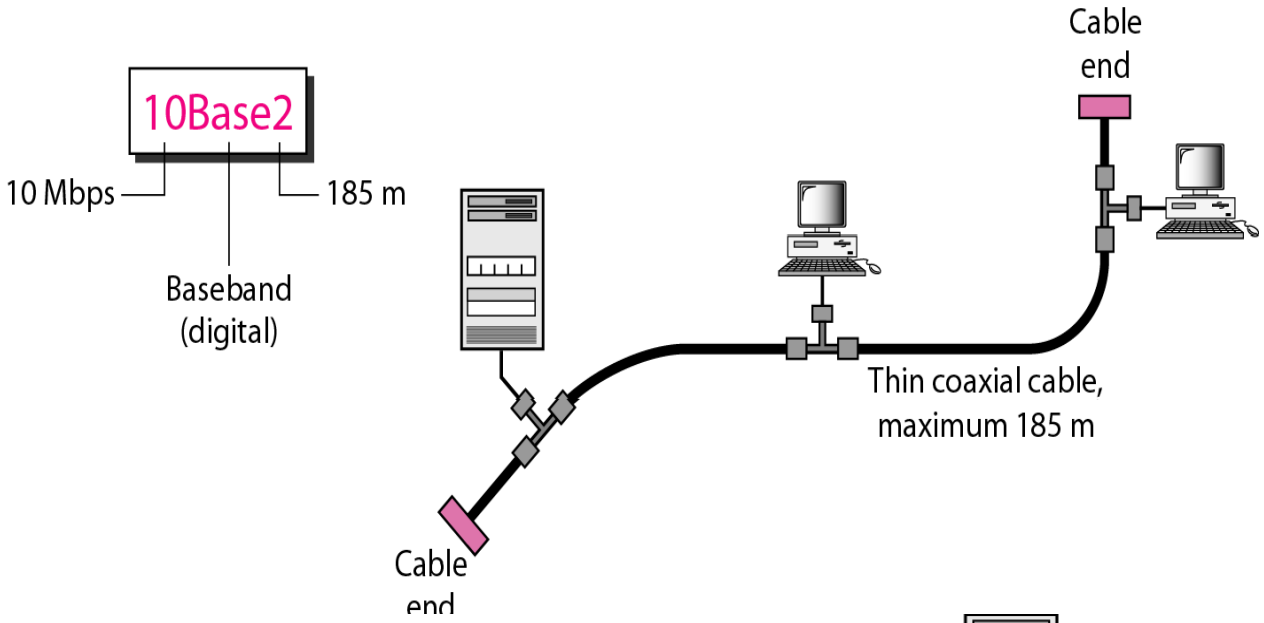
**Broadcast Address** is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is 48-1‟s (all 1‟s in theaddress).

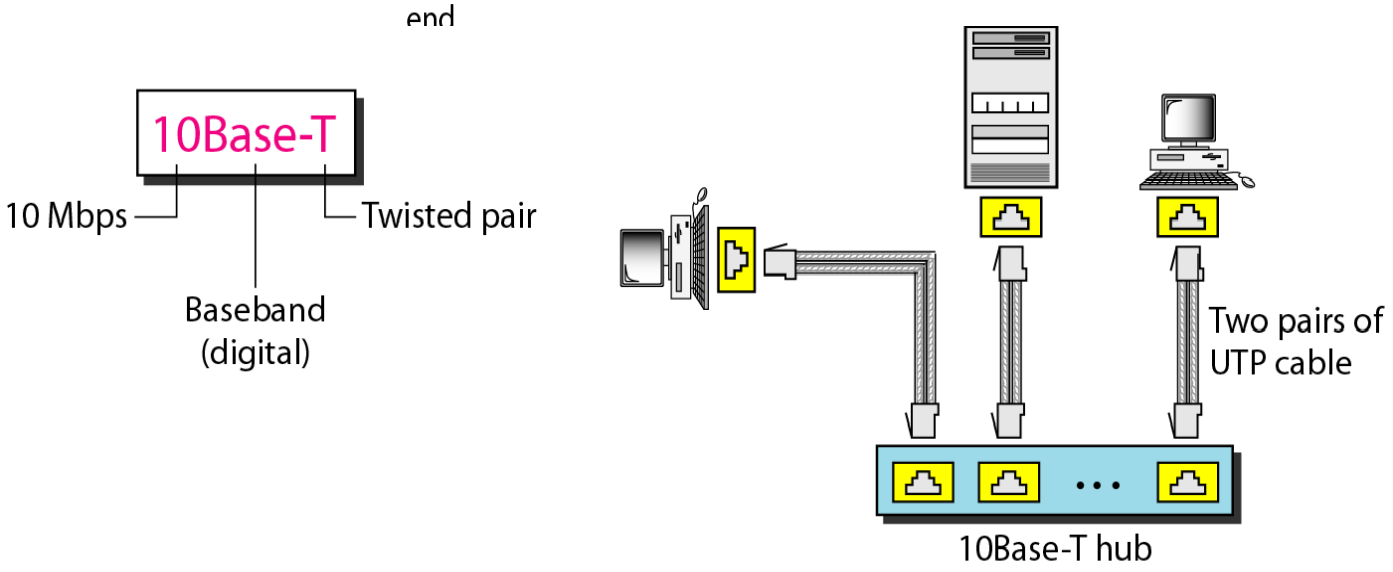
| Slot time = round-trip time + time required to send the jam sequence |
|---|

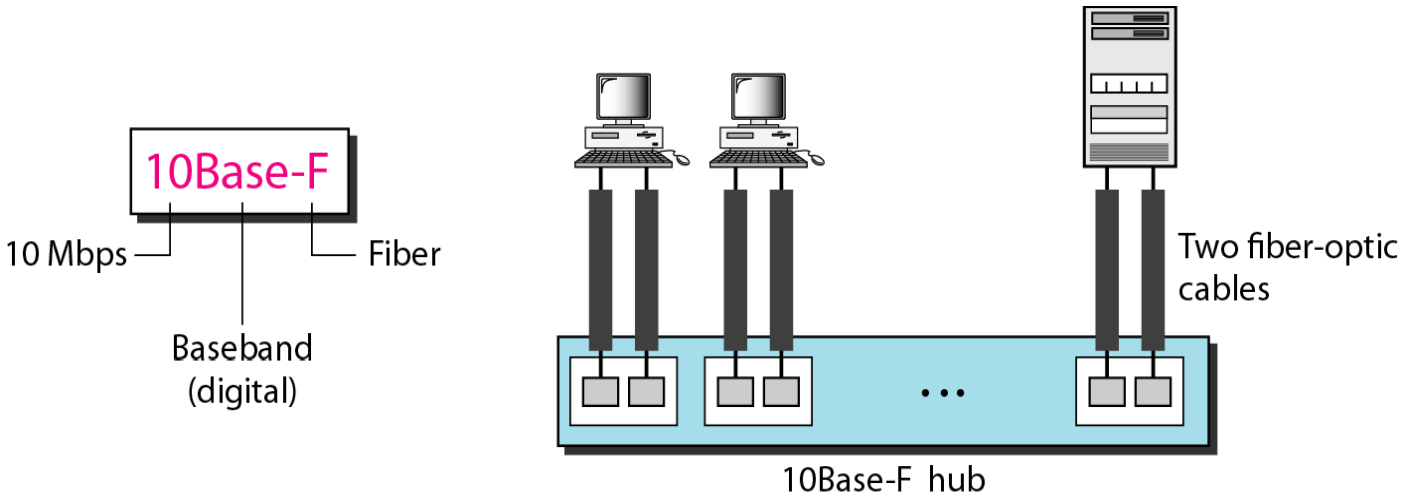
## Categories of Standard Ethernet




10Base2
10 Mbps
185 m
Baseband
(digital)
Cable
end
Thin coaxial cable,
maximum 185 m
Cable
end


10Base-T
10 Mbps
Twisted pair
Baseband
(digital)
Two pairs of
UTP cable
10Base-T hub


10Base-F
10 Mbps
Fiber
Baseband
(digital)
Two fiber-optic
cables
10Base-F hub

## LOGICAL ADDRESSING

Logical addressing is implemented by network layer.

Logical addressing is a Global Addressing scheme.

Data-link layer handles the addressing problem locally, but if packets passes the network boundary there is a need for logical addressing system to help distinguish source and destination systems.

The network layer adds a header to the packet coming from the upper layer that includes the logical addresses of the sender and receiver.

There are 2 types of addressing mechanisms are present:

1. IPv4 ( IPversion4)
2. IPv6 (IP version6)

## IPv4 ADDRESSES

An **IPv4** address is a **32-bit** address that **uniquely** and **universally** defines the connection of a device to the Internet.

**Unique:** Two devices on the Internet can never have the same address at the same time.

**Universal:** The addressing system must be accepted by any host that wants to be connected to the Internet.

### Address Space

- An address space is the total number of addresses used by the protocol.
- If a protocol uses **N** bits to define an address, the address space is $2^N$ because each bit can have two different values (0 or 1) and N bits can have $2^N$ values.
- IPv4 uses **32**-bit addresses, which means that the address space is $2^{32}$ or **4,294,967,296** (more than **4 billion**).

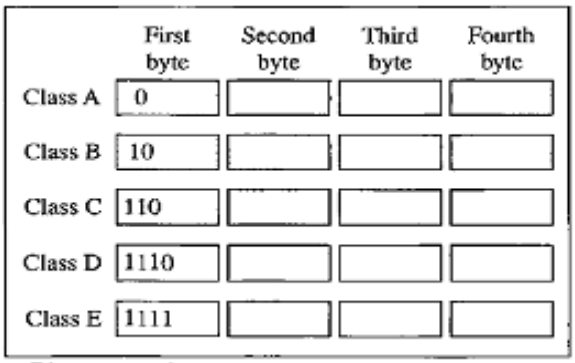
### Notations

There are two notations to show an IPv4 address: Binary and Dotted-Decimal Notation.

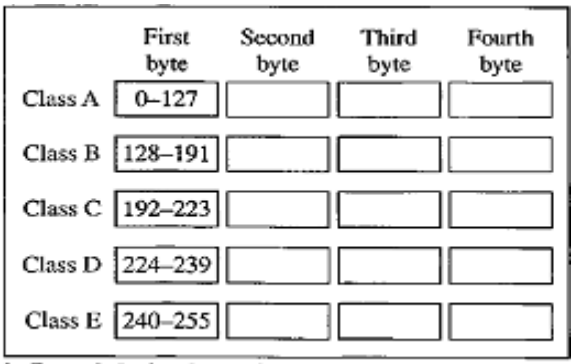
| Binary | Dotted-Decimal |
|---|---|
| • IPv4 address is displayed as 32 bits. Each octet is often referred to as a byte.<br>• It is a 4 byte address<br><br>Ex: 10000000 00001011 00000011 00011111 | • Internet addresses are written in decimal form with a dot separating the bytes.<br>• Each number in dotted-decimal notation is a value ranging from 0 to255.<br><br>Ex: 128.11.3.31 |

## CLASSFUL ADDRESSING

- Initially IPv4 used the concept of Classful addressing.
- In classful addressing, the address space is divided into five classes: A, B, C, D, and E.
- If the address is given in binary notation, the first few bits can immediately tell us the class of the address.
- If the address is given in decimal-dotted notation, the first byte defines the class.



a. Binary notation



b. Dotted-decimal notation

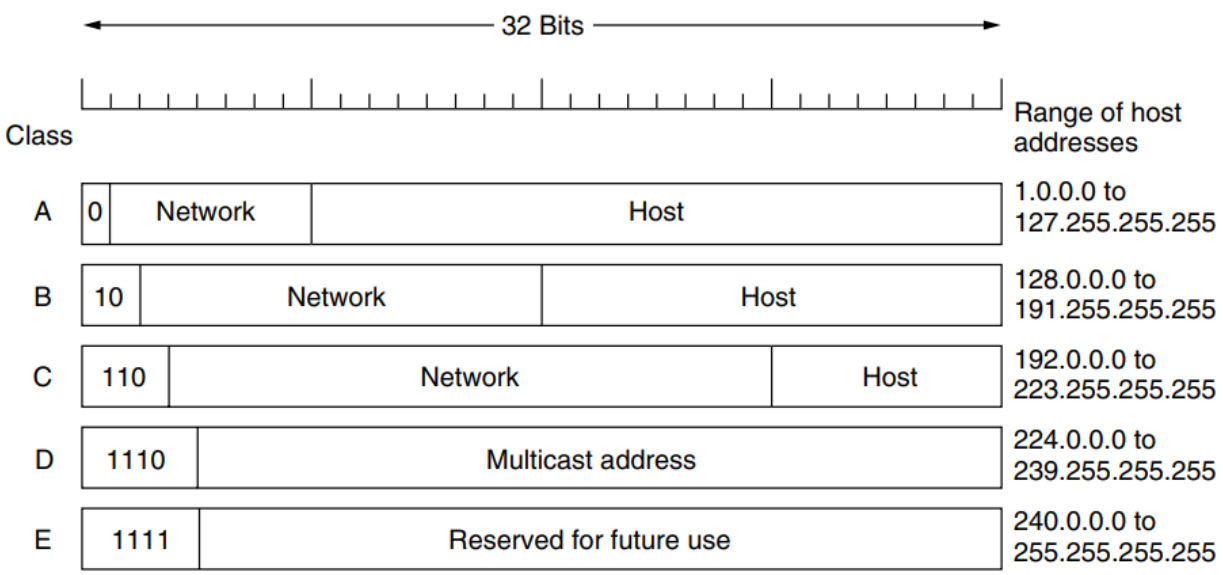
### Classes and Blocks

- Each class is divided into a fixed number of blocks.
- Size of the each block is also fixed.

| Class | No of Blocks | Block Size | Application |
|---|---|---|---|
| A | 128 | 16,777,216 | Unicast |
| B | 16,384 | 65,536 | Unicast |
| C | 2,097,152 | 256 | Unicast |
| D | 1 | 268,435,456 | Multicast |
| E | 1 | 268,435,456 | Reserved |

Purpose of classes:

- **Class A** addresses were designed for large organizations with a large number of attached hosts or routers but most of the addresses in class A were wasted and were not used.
- **Class B** addresses were designed for midsize organizations with tens of thousands of attached hosts or routers.
- **Class C** addresses were designed for small organizations with a small number of attached hosts or routers.
- **Class D** addresses were designed for multicasting, each address in class D is used to define one group of hosts on the Internet.
- **Class E** addresses were reserved for future use.


32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

**Netid and Hostid**

- In classful addressing, an IP address in class A, B, or C is divided into Network id and Host id.

**Mask or Default Mask**

- A default mask is a 32-bit number made of contiguous 1's followed by contiguous0's.
- The mask can help us to find the Netid and the Hostid.
- For example, the mask for a class A address has eight 1s, which means the first 8 bits of any address in class A define the Netid; the next 24 bits define the Hostid.

The masks for classes A, B, and C are:

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

**Subnetting**

- Subnetting is a process of dividing a large block into smaller contiguous groups and assigns each group to smaller networks (subnets) or share a part of the addresses with neighbors.
- Subnetting increases the number of 1's in the mask.

**Supernetting**

- In supernetting, an organization can combine several blocks to create a larger range of addresses. Supernetting decreases the number of 1's in the mask.

Example: an organization that needs 1000 addresses can be granted four contiguous class C blocks. The organization can then use these addresses to create one super network.

**Address Depletion**

- The number of available IPv4 addresses is decreasing as the number of internet users are increasing.
- We have run out of class A and B addresses, and a class C block is too small for most midsize organizations.
- One solution that has alleviated the problem is the idea of **Classless Addressing**.

## CLASSLESS ADDRESSING

**Purpose**

- Classless addressing was designed and implemented to overcome address depletion and give more organizations access to the Internet.
- In this scheme, there are no classes, but the addresses are still granted in blocks.

**Address Blocks**

- In classless addressing, when a small or large entity, needs to be connected to

the Internet, it is granted a block of addresses.

- The size of the block (the number of addresses) varies based on the nature and size of the entity.

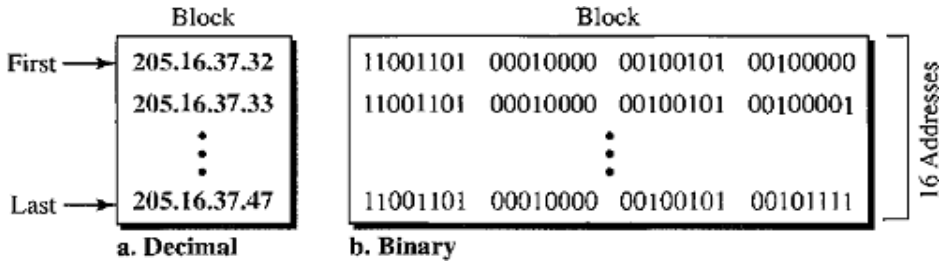
Example:

- For a large organization may be given thousands of addresses
- For a house two addresses are sufficient
- An Internet service provider may be given hundreds of thousands based on the number of customers it may serve.

**Restrictions on classless address blocks**

1. The addresses in a block must be **contiguous**, one after another.
2. The number of addresses in a block must be a power of 2 (1, 2, 4, 8, ...).
3. The first address must be evenly divisible by the number of addresses.

Consider the below figure for classless addressing that shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.




It satisfies all 3 restrictions:

- The addresses are contiguous.
- The number of addresses is a power of 2 (16 =$2^4$).
- The first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in215,024,210.

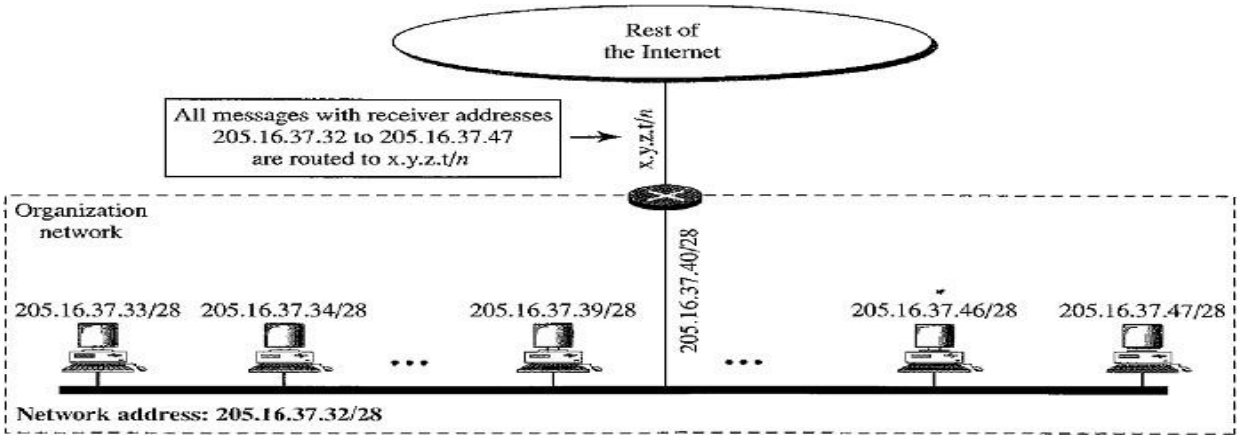
**Mask**

A mask is a 32-bit number in which the ***n*** leftmost bits are 1's and the **32 - *n*** rightmost bits are 0's, where ***n= 0 to32.***

In 1Pv4 addressing, a block of addresses can be defined as **x.y.z.t/n** in which **x.y.z.t** defines one of the addresses and the ***/n*** defines the mask. ***/n*** is called as CIDR notation.

- **First Address** in the block can be found by setting the rightmost **32 - *n*** bits to0's.
- **Last Address** in the block can be found by setting the rightmost **32 - *n*** bits to1's.
- **Number of Addresses** in the block can be found by using the formula $2^{32-n}$.

- **Network Address** is the first address in the block and defines the organization network. Usually the first address is used by routers to direct the message sent to the organization from the outside.



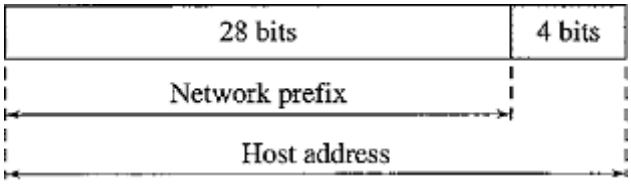

Example: **205.16.37.39/28** or **11001101 00010000 00100101 00100111**

- First address: 11001101 000100000100101 0010000 or 205.16.37.**32**
- Last address: 11001101 00010000 001001010010 1111 or205.16.37.**47**
- Number of Addresses: $\mathbf{2^{32-28} = 2^4=16.}$
- Network Address (First Address) 11001101 000100000100101 0010000 or205.16.37.32

**Netid and Hostid**

- The ***n*** leftmost bits of the address **x.y.z.t/n** define the **network address** or **prefix**.
- The **(32 – *n*)** rightmost bits define the particular **suffix** or **host address** (computer or router) connected to the network.

**205.16.37.39/28** or **11001101 00010000 00100101 0010 0111**



## Network Address Translation (NAT)

- As the number of home users and small business users are increasing day by day it is not possible to give each and every user to one IPv4 address due to shortage of IPv4 addresses.
- In order to overcome this problem the developers designed the concepts of private IP address and Network Address Translation(NAT).
- **NAT** enables a user to have a large set of addresses internally (private IP addresses) and a small set of addresses externally (public IP addresses).

The Internet authorities have reserved three sets of addresses as private addresses:

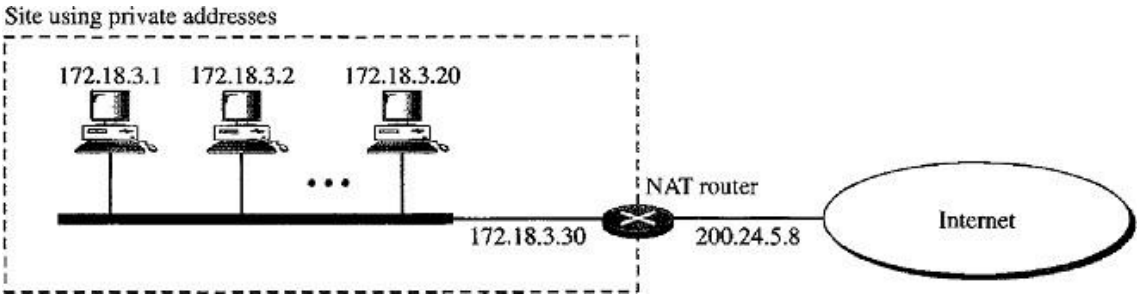
| Range | Total |
|---|---|
| 10.0.0.0 to 10.255.255.255 | $2^{24}$ |
| 172.16.0.0 to 172.31.255.255 | $2^{20}$ |
| 192.168.0.0 to 192.168.255.255 | $2^{16}$ |

Any organization can use an address out of this set without permission from the Internet authorities.

- They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address.

Example:

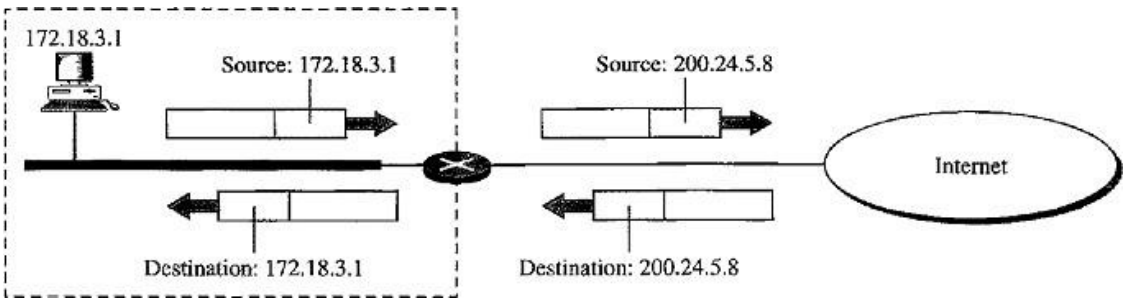
Consider the below figure describes the private network with private addresses:




- The NAT router has one public address **200.24.5.8,** it is a global address.
- The internal devices having addresses from **172.18.3.1** to **172.18.3.30** these are local addresses.

**Address Translation**

- Alltheoutgoingpacketsgothroughthe NATrouter,whichreplacesthe**sourceaddress** in the packet with the **global** NAT address.
- All incoming packets also pass through the NAT router, which replaces the



**destination address** in the packet (the NAT router global address) with the appropriate **private address.**

**Translation Table**

There are two types of translation tables:

1. Two Column translation table (Using one IP address)
2. Five column translation table (Using IP addresses and Port Numbers)

**Two Column Translation Table**

- It contains two columns: Private Address and External Address.
- In this strategy, communication must always be initiated by the private network.
- When the router translates the source address of the outgoing packet, it also makes note of the destination address-where the packet is going.
- When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet.

**Translation Table**

| Private | External |
|---|---|
| 172.18.3.1 | 25.8.2.10 |
| … | …. |
| …. | …. |

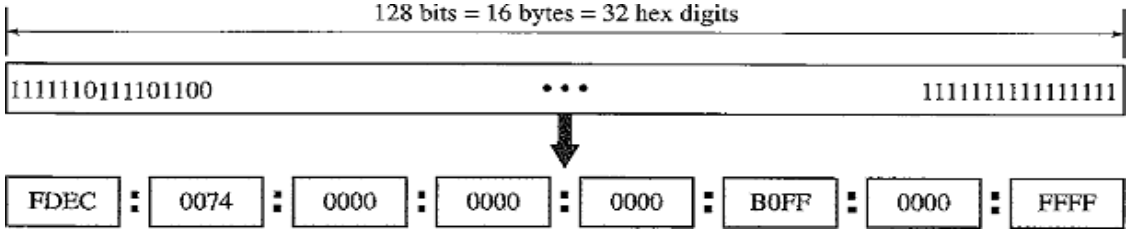
**IPv6 ADDRESSES (Internetworking Protocol version6)**

Why IPv6?

- In order to overcome the problems of address depletion.
- It eliminates the concept of NAT and Private Addresses.
- There is no need for classless addressing and DHCP.

**Structure**

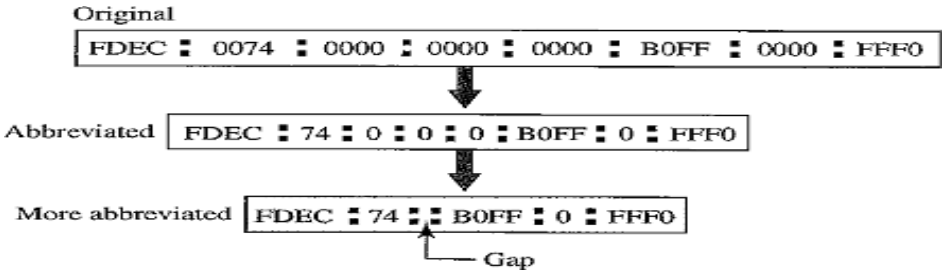
IPv6 specifies hexadecimal colon notation (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).
An IPv6 address consists of 16 bytes (octets); it is 128 bits long. 128 bits is divided into eight sections. Each of 4 hex digits separated by a colon.




**Abbreviation**

- Hexa decimal format of IPv6 is very long and many of the digits are zeros.
- We can abbreviate this address as the leading zeros of a section (four digits between two colons) can be omitted.

Note: Only leading zeros are omitted not
trailing zeros. Example:



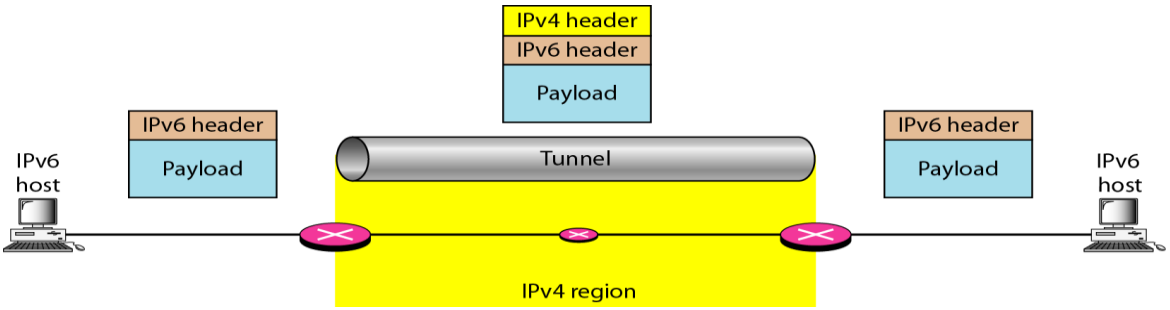

- In the above example: 0074 written as 74, 0000 written as 0.
- If there are consecutive sections consisting of zeros only. We can remove the zeros altogether and replace them with a double semicolon.

## Address Space

- IPv6 has $2^{128}$ addresses are available. It is a much larger address space than IPv4

## Tunneling

- Tunneling is a strategy used when two computers using IPv6 want to communicate with each other and the packet must pass through a region that usesIPv4.
- To pass through this region, the packet must have an IPv4address.
- So the IPv6 packet is encapsulated in an IPv4 packet when it enters the region, and it leaves its capsule when it exits the region.
- It seems as if the IPv6 packet goes through a tunnel at one end and emerges at the other end. The IPv4 packet is carrying an IPv6 packet as data, the protocol value is set to41.



## IPv4 Delivery Mechanism

- IPv4 delivery mechanism is used in TCP/IP protocols.
- IPv4 is an unreliable and connectionless datagram protocol.
- If reliability is important, IPv4 must be paired with a reliable protocol such as TCP.

**Datagram**

Packets in the IPv4 layer are called datagrams.

A datagram is a variable-length packet consisting of two parts:

1. Header
2. Data

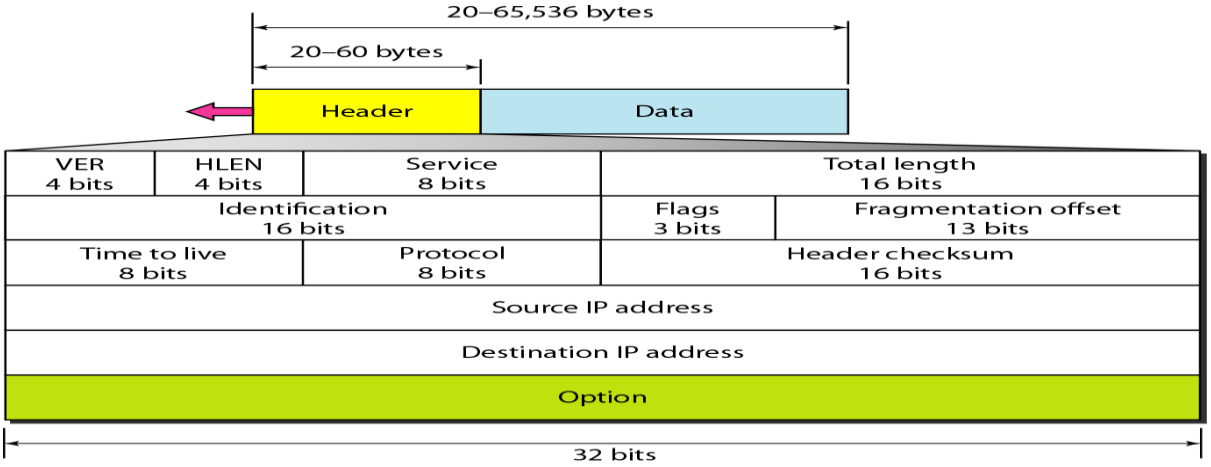
The header is 20 to 60 bytes in length and contains information essential to routing and delivery.

**1. Version (VER) – 4bits**

- It defines the version of the IPv4 protocol. Currently 4th version of IPv4 is using.

**2. Header length (HLEN) – 4bits**

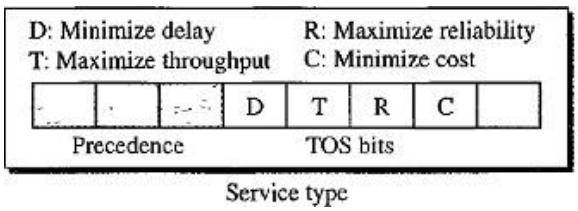
- It defines the total length of the datagram header in 4-bytewords.
- The length of the header is variable between 20 and 60bytes.
- When there are no options, the header length is 20 bytes, and the value of this field is 5 (5x 4 =20).
- When the option field is at its maximum size, the value of this field is 15 (15 x 4 = 60).



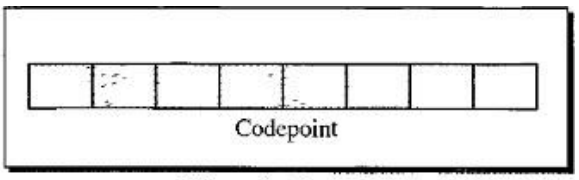

**3. Services**

IETF has changed the interpretation and name of this 8-bit field.

Previously this field is called **Service Type** but now the name changed to **Differentiated Services**.




Service type




Differentiated services

**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.

**i. Precedence**

- It is a 3-bit subfield ranging from 0 to 7 (000 to111).
- The precedence defines the priority of the datagram in issues such as congestion.
- If a router is congested and needs to discard some datagrams, those datagrams with lowest precedence are discarded first.

**ii. Type of Service(TOS)**

It is a 4-bit subfield with each bit having a special meaning. Out of 4 bits only one bit will have the value of 1.

| TOS Bits | Description |
|---|---|
| 0000 | Normal (default) |
| 0001 | Minimize Cost |
| 0010 | Maximize Reliability |
| 0100 | Maximize Throughput |
| 1000 | Minimize Delay |

**Differentiated Services**

In this interpretation, the first 6 bits make up the code-point subfield, and the last 2 bits are not used. The code-point subfield can be used in two different ways:

a. When the 3 rightmost bits are 0's, the 3 leftmost bits are interpreted the same as the precedence bits in the service type interpretation.

b. When the 3 rightmost bits are not all Os, the 6 bits define 64 services based on the priority assignment by the Internet or local authorities.

| **Category** | **Code-point** | **Assigning Authority** | **No of service types** | **Numbers** |
|---|---|---|---|---|
| 1 | XXXXX0 | Internet | 32 | 0,2,4,6,8,……60,62 |
| 2 | XXXX11 | Local | 16 | 3,7,11,15,……59,63 |
| 3 | XXXX01 | Temporary or Experiment | 16 | 1,5,9,13,17...……,61 |

**4. Total length**

This is a 16-bit field that defines the total length (**header plus data**) of the IPv4 datagram in bytes. Total length of IPv4 is 65,535 ($2^{16}$-1).

Length of data = Total length - header length

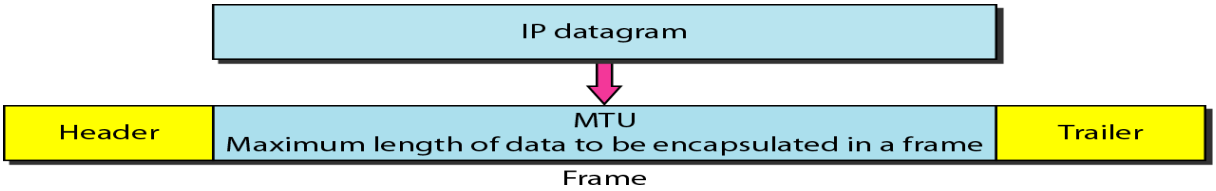
**FRAGMENTATION**

- A datagram is fragmented if it is too large for a network to carry it.

**Maximum Transfer Unit (MTU)**

- the Total size of the datagram must be less than this maximum size.

- The maximum length of IPV4 datagram is 65,535bytes.




- If the length of the datagram exceeds the MTU then the datagram must be fragmented to make it possible to pass through the networks.
- When a datagram is fragmented, each fragment has its own header with only few fields are changed. Remaining fields are copied by all fragments.

**Fields Related to Fragmentation: Identification, Flag, Offset.**

**Identification (16 bits)**

- When a datagram is fragmented, all fragments have the same identification number the same as the original datagram. All fragments having the same identification value must be assembled into one datagram.
- The identification number helps the destination in reassembling the datagram.

**Flags (3 bits)**

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 its value is 0, the datagram can be fragmented if necessary.

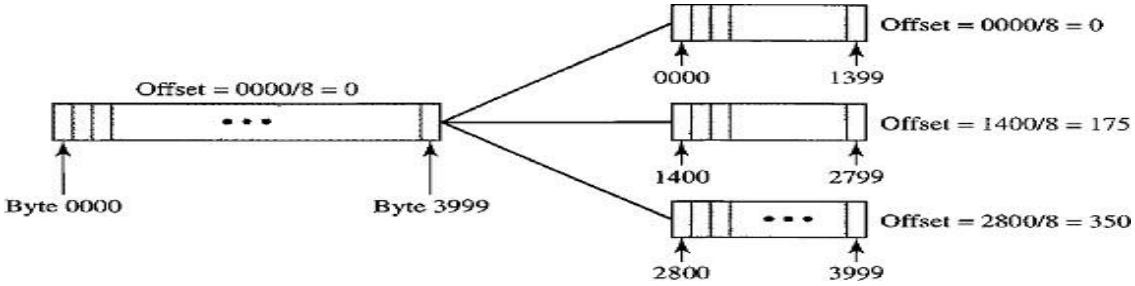
The third bit is called the **More Fragment** bit.

- If its value is 1, it means the datagram is not the last fragment.
- If its value is 0, it means this is the last or only fragment.

**Fragmentation offset (13 bits)**

It 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.

Example: Consider the below figure shows a datagram with a data size of 4000 bytes fragmented into three fragments.




The bytes in the original datagram are numbered 0 to 3999.

| Fragment Number | Range | Offset Value |
|---|---|---|
| First | 0-1399 | 0/8=0 |
| Second | 1400-2799 | 1400/8=175 |
| Third | 2800-3999 | 2800/8=350 |

**Time To Live - TTL (8 bits)**

A datagram has a limited lifetime in its travel through an internet. This field can be used in two ways:

i. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero.

ii. This field is used mostly to control the maximum number of hops (routers) visited by the datagram. Each router that processes the datagram decrements this number by 1. The router discards the datagram, if **TTL=0**.
When a source host sends the datagram, it stores a number in TTL field. This value is approximately 2 times the maximum number of routes between any two hosts.

**Protocol (8 bits)**

- This field defines the higher-level protocol that uses the services of the IPv4layer.
- 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.

**Checksum (16 bits)**

The checksum in the IPv4 packet covers only the header, not the data. There are two reasons:

i. All higher-level protocols that encapsulate data in the IPv4 datagram have a checksum field that covers the whole packet. The checksum for the IPv4 datagram does not have to check the encapsulated data.

ii. The header of the IPv4 packet changes with each visited router, but the data do not changes. So the checksum includes only the part that has changed.

**Options**

Options are not required for a datagram. They can be used for network testing and debugging.

**End of Option**: An end-of-option option is a 1-byte option used for padding at the end of the option field.
**Record Route**: A record route option is used to record the Internet routers that handle the datagram.
It can list up to nine router addresses. It can be used for debugging and management purposes.

**Strict Source Route**: A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput.
If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued.

**Loose Source Route**: A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

**Timestamp**: A timestamp option is used to record the time of datagram processing by a router.

**Source Address (32 bits) & Destination Address (32 bits)**

- These two fields define the IPv4 address of the Source and Destination respectively.
- These fields must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

**Disadvantages of IPv4**

1. Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
2. The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
3. The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided byIPv4.

## IPV6 DELIVERY MECHANISM

- IPV6 is introduced to overcome the deficiencies ofIPv4.
- IPv6 is also called as IPng (Internetworking Protocol next generation).
- In IPv6, the Internet protocol was extensively modified to accommodate the growth of the Internet. Packet format, Length of IP address, ICMP, IGMP, ARP, RARP, RIP routing protocol are also modified inIPv6.

**Advantages of IPv6**

- **Larger address space** An IPv6 address is 128 bits long whereas IPv4 is 32-bitaddress.
- **Better header format** IPv6 uses a new header format in which options are separated from the base header. when options are needed it is inserted between the base header and the upper-layer data.
- **New options** IPv6 has new options to allow for additional functionalities.
- **Allowance for extension** IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- **Resource Allocation** In IPv6 the type-of-service field has been removed, but a mechanism called ***flow label*** has been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- **More Security** The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

**Packet Format**

In IPv6 each packet is composed of a mandatory base header followed by the payload.
The **Payload** consists of two parts:

- Optional extension headers
- Data from an upper layer

The **Base Header** occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.

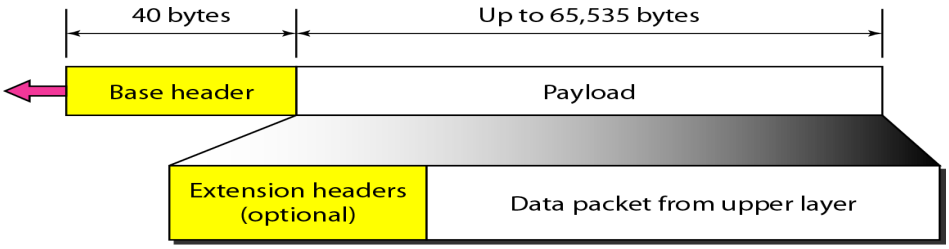
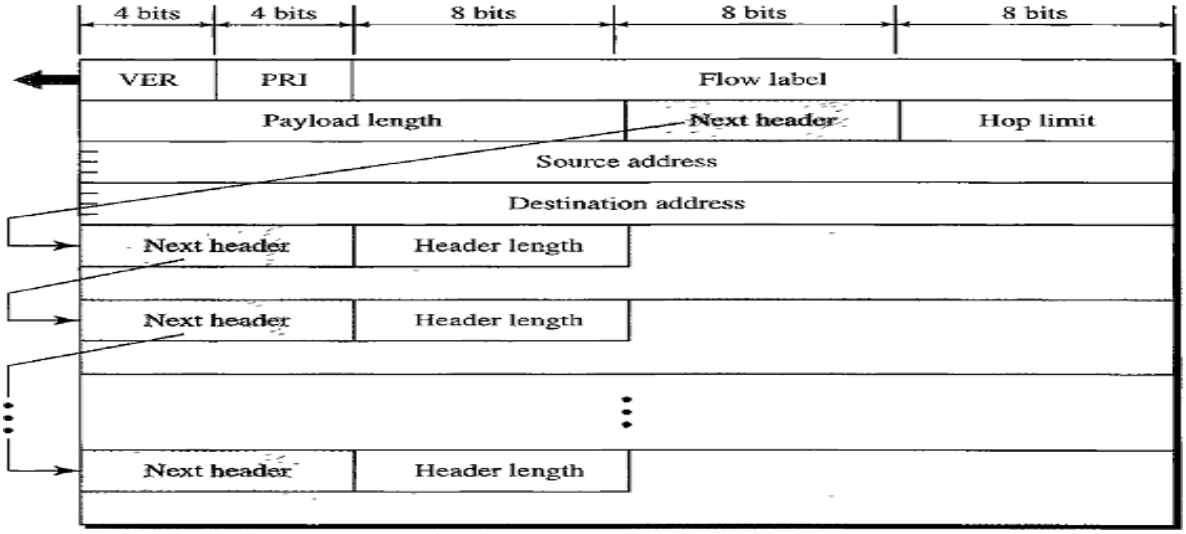



**Fig: IPv6 Datagram header and payload**

**Base Header**

Fields in IPv6 datagram are:




- **Version (4-bit)**

  This field defines the version number of the IP. For IPv6, the value is 6.

- **Priority(4-bit)**

  The priority field defines the priority of the packet with respect to traffic congestion.

- **Flow label (24-bit or 3Byte)**

  Flow label field that is designed to provide special handling for a particular flow of data.

- **Payload length (16 bit or 2Byte)**

  Payload length field defines the length of the IP datagram excluding the base header.

- **Next header(8-bit)**

  The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either optional extension headers used by IP or the header of TCP or UDP encapsulated packet.

  **Note**: This field in IPv4 is called the *protocol.*

- **Hop limit (8 bit)**

  Hop limit field serves the same purpose as the TTL field in IPv4

- **Source address (128-bit or 16 Byte) and Destination Address (128 bit or 16Byte)**

  The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.

  The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram. If source routing is used, this field contains the address of the next router.

Next Header codes for IPv6:

| Code | Next Header |
|---|---|
| 0 | Hop-by-hop option |
| 2 | ICMP |
| 6 | TCP |
| 17 | UDP |
| 43 | Source Routing |
| 44 | Fragmentation |
| 50 | Encrypted security payload |
| 51 | Authentication |
| 59 | Null (No Next Header) |
| 60 | Destination Option |

**Priority**

The priority field of the IPv6 packet defines the priority of each packet with respect to other packets from the same source.

Example: If one of two consecutive datagrams must be discarded due to congestion, the datagram with the lower **packet priority** will be discarded.

IPv6 divides traffic into two broad categories:

i. Congestion-Controlled
ii. Non congestion-controlled

**Congestion-Controlled Traffic**

When there is congestion a source adapts itself to slowdown the traffic.

Example: TCP uses sliding window protocol can easily respond to the traffic.

Congestion-controlled data are assigned priorities from 0 to 7

| Priority | Meaning | Description |
|---|---|---|
| **0** | No specific traffic | Priority 0 is assigned to a packet when the process does not define a priority. |
| **1** | Background data | defines data that are usually delivered in the background.<br>Ex: Delivery of the news. |
| **2** | Unattended data traffic | If the user is not waiting (attending) for the data to be received, the packet will be given a priority of 2.<br>Ex: Email |
| **3** | Reserved | |
| **4** | Attended bulk data traffic | A protocol that transfers data while the user is waiting to receive the data is given a priority of 4<br>Ex: FTP and HTTP |
| **5** | Reserved | |
| **6** | Interactive traffic | Protocols that need user interaction are assigned 6.<br>Ex: TELNET |
| **7** | Controlled traffic | Routing Protocols are given Highest Priority 7.<br>Ex: OSPF, RIP, SNMP |

**Non congestion-Controlled Traffic**

- The source does not adapt itself to congestion. It is a type of traffic that expects minimum delay. Priority numbers from 8 to 15 are assigned to Non congestion-controlled traffic.
- In this traffic Discarding of packets is not desirable and Retransmission in most cases is impossible.

Examples: Real-time audio and video.

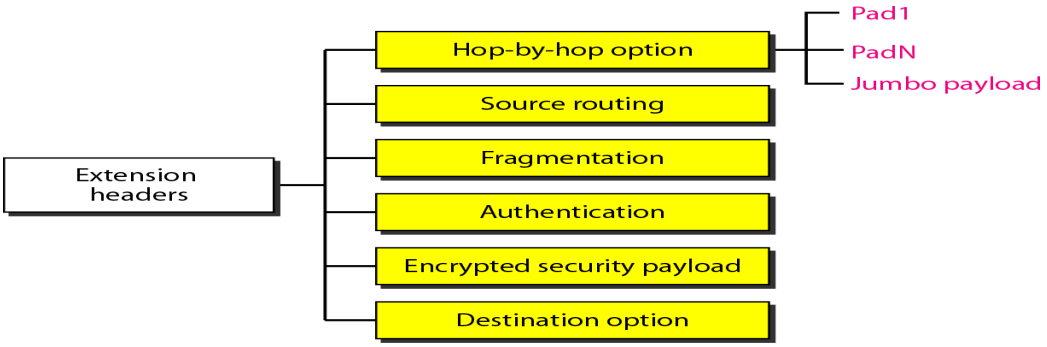
- **Priority 15:** It is given to data containing **Less Redundancy** (low-fidelity audio or video)
- **Priority 8:** It is given to data containing M**ore Redundancy** (high-fidelity audio or video)

**Flow Label**

- A sequence of packets, sent from a particular source to destination that needs special handling by routers is called a F**low of packets.**
- The combination of the source address and the value of the **Flow Label** uniquely define a flow of packets.
- To a router, a flow is a sequence of packets that share the same characteristics such as traveling the same path, using the same resources, having the same kind of security etc.
- A router that supports the handling of flow labels has a flow label table. The table has an entry for each active flow label.
- Each entry defines the services required by the corresponding flow label.
- When a router receives a packet it consults the flow label table instead of consulting the routing table and going through a routing algorithm to define the address of the next hop, it can easily look in a flow label table for the next hop.
- This mechanism speed up the processing of a packet by a router.

**Extension Headers**

To give greater functionality to the IP datagram, the base header can be followed by up to six types of extension headers.




**Hop-by-Hop Option**

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram.

Only three options have been defined: Pad l, Pad N, and jumbo payload.

- The Pad l option is 1 byte long and is designed for 1 byte alignment purposes.
- Pad N is used when 2 or more bytes is needed for alignment.
- The jumbo payload option is used to define a payload longer than 65,535bytes.

**Source Routing**

- The source routing extension header combines the concepts of the strict source route and the loose source route options ofIPv4.

**Fragmentation**

- In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels.
- In IPv6, only the original source can fragment. A source must use a path MTU discovery technique to find the smallest MTU supported by any network on the path.
- The source then fragments using this knowledge.

**Authentication**

- The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data.

**Encrypted Security Payload (ESP)**

- ESP is an extension that provides confidentiality and guards against eavesdropping.

**Destination Option**

- It is used when the source needs to pass information to the destination only.
- Intermediate routers are not permitted access to this information.

**Comparison between IPv4 Options and IPv6 Extension Headers**

1. The no-operation and end-of-option options in IPv4 are replaced by Pad1 and Pad N options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header inIPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header inIPv6.
6. The authentication and Encrypted Security Payload extension headers are new inIPv6.

# ADDRESS MAPPING

The internet is made of a combination of physical networks connected by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host. The hosts and routers are recognized at the **network level** by their **logical (IP) addresses**, while at the **physical level**, theyare recognized by their **physical (MAC) addresses.**

Thus delivery of a packet to a host or a router requires **two levels** of **addressing**: **logical (IP)** and **physical (MAC)**.

**We need to be able to map a logical address to its corresponding physical address and vice-versa.These can be done by using either static or dynamic mapping.**

### Static mapping

Static mapping involves the creation of a table that associates a logical address with a physical address.

**Limitations:**

- A machine could change its NIC (Network Interface Card), resulting in a new physicaladdress.
- In some LANs, such as LocalTalk, the physical address changes every time the computeris turned on.
- A mobile computer can move from one physical network to another, resulting in achange in its physical address.

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

### Dynamic mapping

In such mapping each time a machine knows one of the two addresses (logical or physical), itcan use a protocol to find the other one.

### Mapping Logical to Physical Address: ARP

ARP stands for **Address Resolution Protocol** which is one of the most important protocols of the Network layer in the OSI model. ARP finds the physical address, also known as Media Access Control (MAC) address, of a host from its known IP address Figure 21.2.

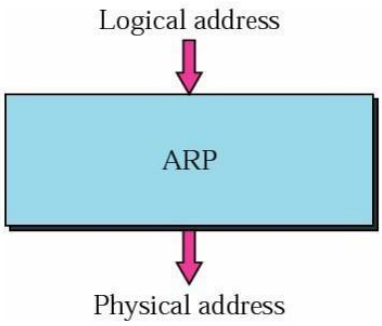



Figure 21.2: ARP Mapping

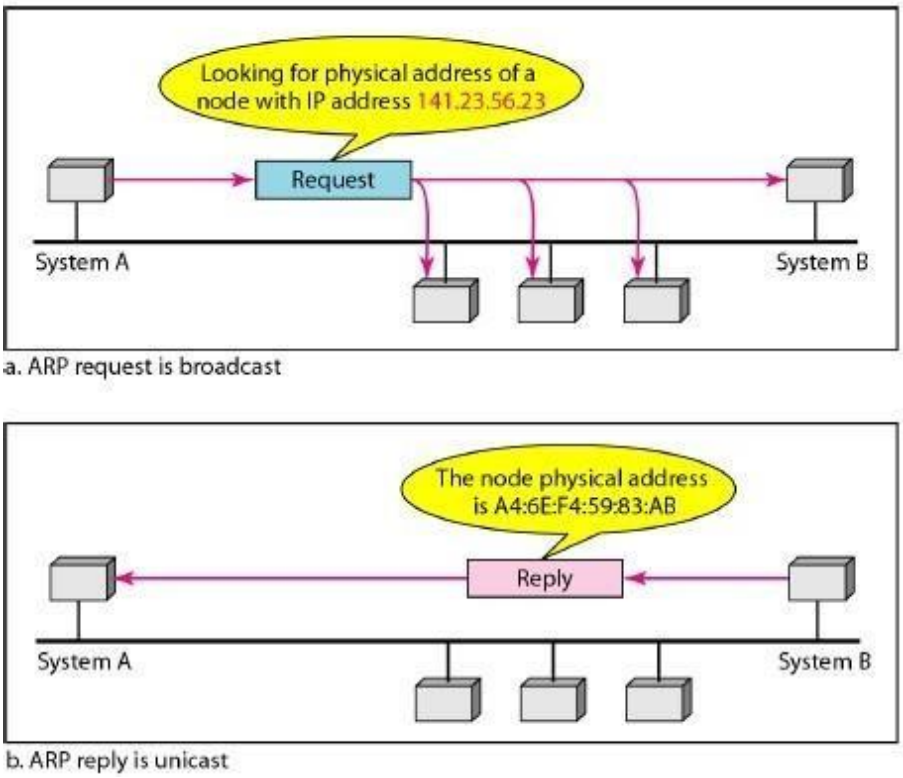



Figure 21.3 ARP operation

Following **steps** are involved in logical to physical address mapping:

**a.** The host or the router sends an **ARP query packet**. The ARP query packet includes the physical and IP addresses of the sender and the IP address of the receiver. As the sender does not know the physical address of the receiver, the **ARP query is broadcast over the network** (see Figure 21.3).

**b.** Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an **ARP response packet**.

**c.** The ARP response packet contains the recipient's IP and physical addresses. The ARP response packet is **unicast directly to the inquirer** (host/router) by using the physical address received in the query packet.

**Example**: (Figure 21.3)The system on the left (A) has a packet that needsto be delivered to another system (B) with IP address 141.23.56.23.
System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of 141.23.56.23. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 21.3b.
System B sends an ARP reply packet that includes its physical address.

Now system A can send all the packets it has for this destination by using the physical address it received.

**Cache Memory**

Using ARP is inefficient if system A needs to broadcast an ARP request for each IP packet it needs to send to system B. ARP can be useful if the ARP reply is cached (kept in cache memory for a while) because a system normally sends several packets to the same destination. A system that receives an ARP reply stores the mapping in the cache memory and keeps it for 20 to 30 minutes unless the space in the cache is exhausted. Before sending an ARP request, the system first checks its cache to see if it can find the mapping.

**ARP Packet Format**

Figure 21.4 shows the format of an ARP packet.

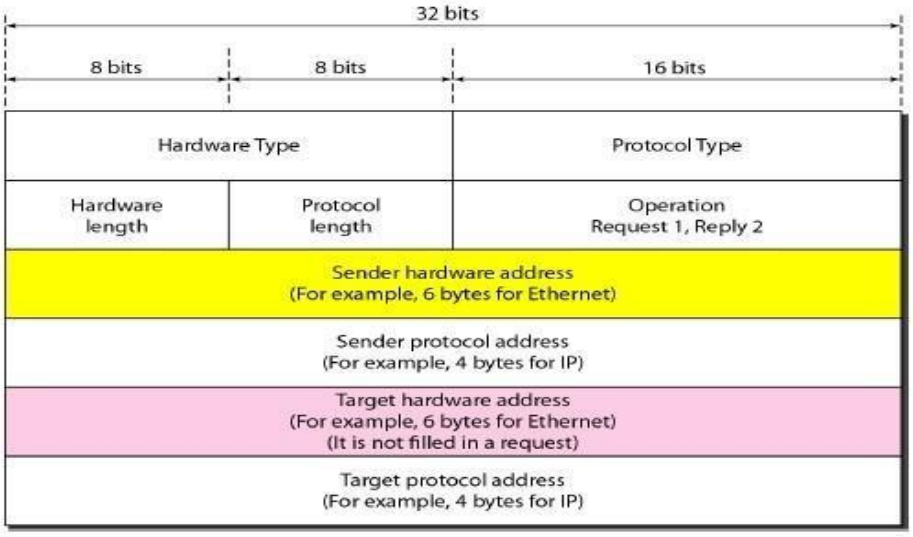



Figure 21.4 ARP packet

The fields are as follows:

a. **Hardware type.** This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For **example**, Ethernet is given type 1. ARP can be used on any physical network.

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

c. **Hardware length**. This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.

d. **Protocol length**. This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.

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

f. **Sender hardware address**. This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.

g. **Sender protocol address**. This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.

h. **Target hardware address**. This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is alI 0s because the sender does not know the physical address of the target.

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

**Encapsulation**

An ARP packet is encapsulated directly into a data link frame. For example, in Figure 21.5 an ARP packet is encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame are an ARP packet.

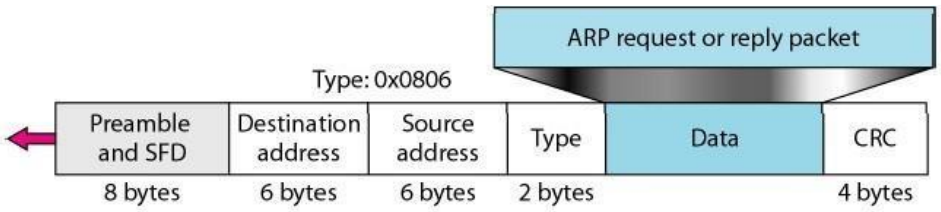



Figure 21.5 Encapsulation of ARP packet

**ARP Operation**

Let us see how ARP functions on a typical internet. First we describe the steps

involved. Then we discuss the four cases in which a host or router needs to use ARP. These are the steps involved in an ARP process:

**1.** The sender knows the IP address of the target. We will see how the sender obtains thisshortly.

**2.** IP asks ARP to create an ARP request message, filling in the sender physical address, thesender IP address, and the target IP address. The target physical address field is filled with 0s.

**3.** The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.

**4.** Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.

**5.** The target machine replies with an ARP reply message that contains its physical address. The message is unicast.

**6.** The sender receives the reply message. It now knows the physical address of the targetmachine.

**7.** The IP datagram, which carries data for the target machine, is now encapsulated in a frameand is unicast to the destination.

## Proxy ARP

A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router. Let us give an example.In Figure 21.8 the ARP installed on the right-hand host will answer only to an ARP request witha target IP address of 141.23.56.23.

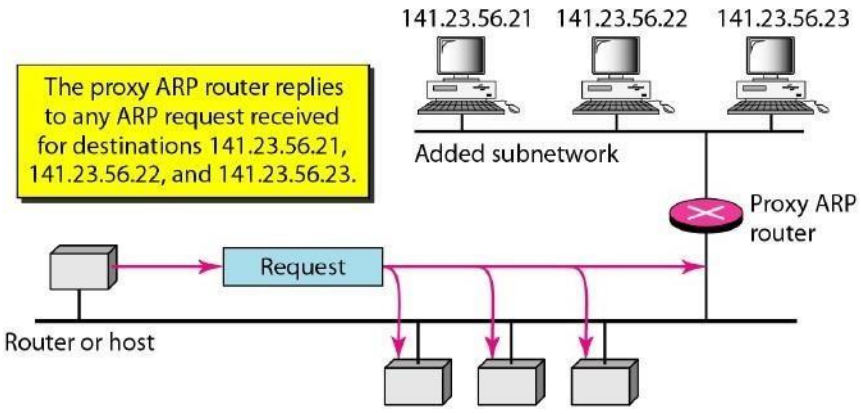



Figure 21.8 Proxy

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

## Mapping Physical to Logical Address: RARP, BOOTP, and DHCP

There are occasions in which a **host knows its physical address, but needs to know its logical address** Figure 21.9. This may happen in **two cases**:

**Case 1:** *A diskless station is just booted*. The station can find its physical address by checking its interface, but it does not know its IP address.

**Case 2:** *An organization does not have enough IP addresses to assign to each station*; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

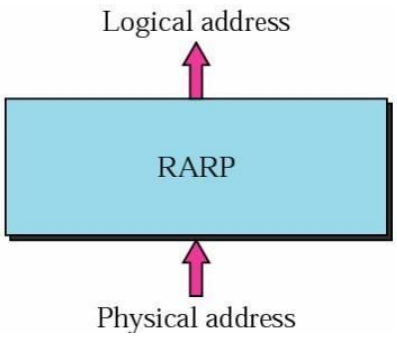


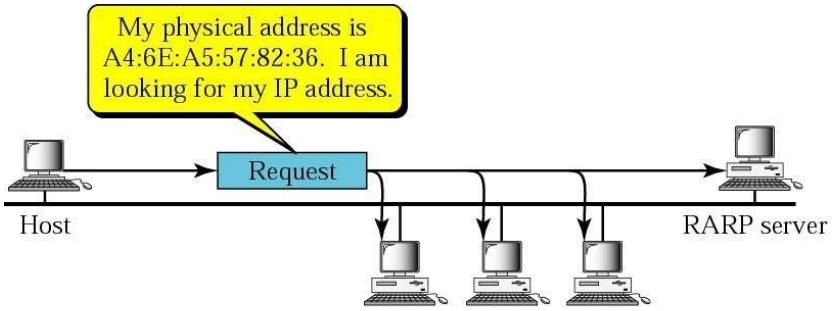

Figure 21.9: RARP Mapping

## RARP

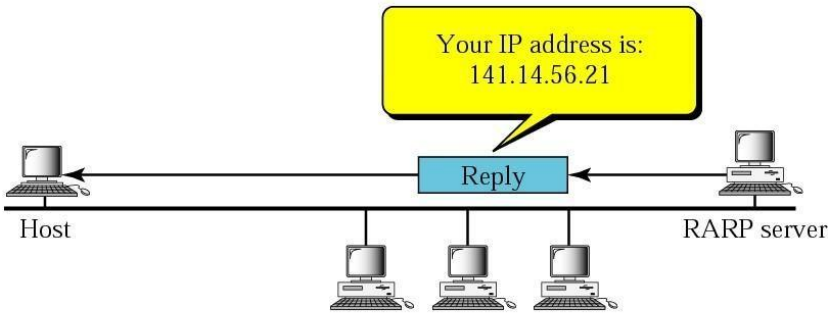
**Reverse Address Resolution Protocol** (RARP) finds the logical address for a machine that knows only its physical address. RARP Operation

RARP operation is displayed in Figure 21.10

a. A **RARP request** is created and **broadcast** on the local network.
b. Another machine on the local network that knows all the IP addresses will respond with a **RARP reply**.
c. The requesting machine must be running a **RARP client** program; the responding machine must be running a **RARP server** program.




a. RARP request is broadcast




b. RARP reply is unicast

Figure 21.10 RARP Operation

**RARP Packet Format & Encapsulation**

The format of the RARP packet is the same as the ARP packet format as displayed in Figure 21.4, except that the Operation field. It's value is 3 for RARP request message and 4 for RARP reply message.

An RARP packet is also encapsulated directly into a data link frame just like ARP packet as displayed in Figure 21.5.

**Limitations of RARP:**

- As broadcasting is done at the data link layer. The physical broadcast address, all 1's inthe case of Ethernet, does not pass the boundaries of a network.
- This means that if an administrator has several networks or several subnets, it needs toassign a RARP server for each network or subnet.
- This is the reason that **RARP is almost obsolete**.
- Two protocols, BOOTP and DHCP, are replacing RARP.

# BOOTP

The Bootstrap Protocol (BOOTP) is a client/server based protocol at application layer, designed to **provide physical address** to **logical address mapping**. The administrator may put the client and the server on the same network or on different networks, as shown in Figure 21.11a and Figure 21.11b respectively. **BOOTP** messages are **encapsulated** in a **UDP packet**, and the UDP packet itself is encapsulated in an **IP packet**, as shown in Figure 21.12.

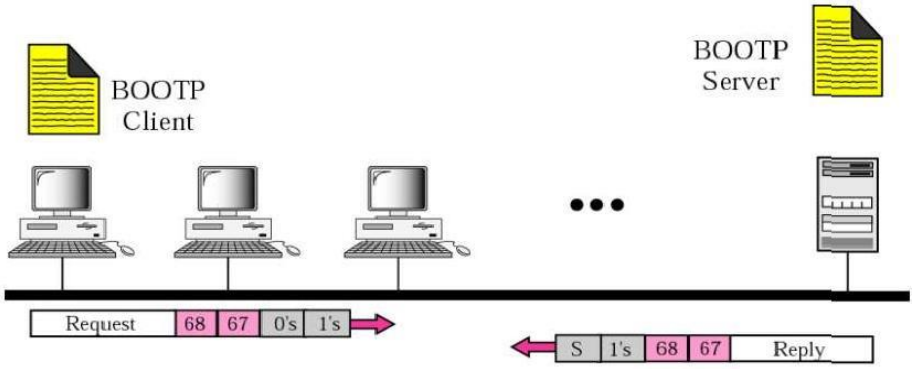



Figure 21.11a BOOTP client and server on the same network

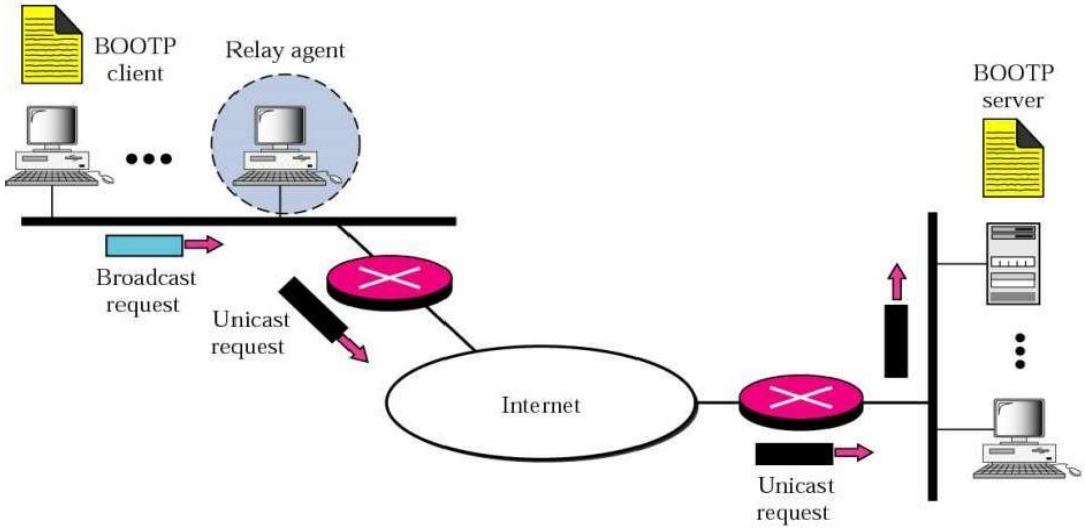



Figure 21.11b BOOTP client and server on the same and different network

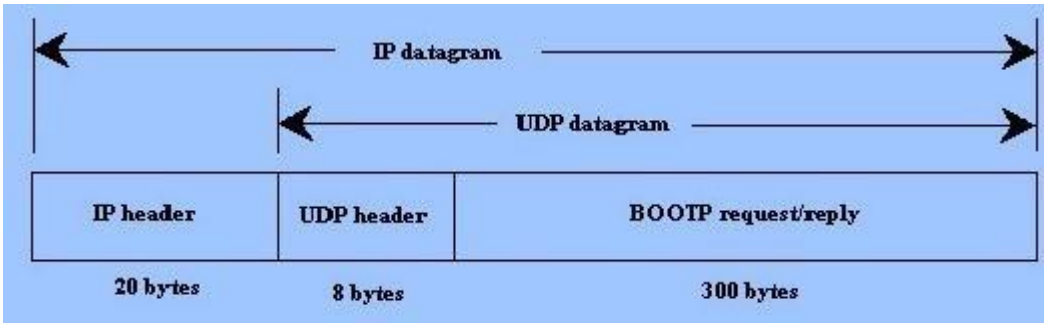



Figure 21.12 BOOTP data Encapsulation

**BOOTP Operation**

There are two cases of BOOTP operation described below:

***Case 1: Client and server on same network*** (*Figure 21.11a)*

1. When a BOOTP client is started, it has no IP address, so it broadcasts a message containing its MAC address onto the network. This message is called a "BOOTP request," and it is picked up by the BOOTP server, which replies to the client with the following information that the client needs:
    a. The client's IP address, subnet mask, and default gateway address.
    b. The IP address and host name of the BOOTP server.
    c. The IP address of the server that has the boot image, which the client needs to load its operating system.
2. When the client receives this information from the BOOTP server, it configures and initializes its TCP/IP protocol stack, and then connects to the server on which the boot image is shared.

**Case 2 : Client and server on different networks(**Figure 21.11b**)**

1. If the server exists on some distant network the BOOTP request is broadcast because the client does not know the IP address of the server.
2. The client simply uses all as 0's the source address and all 1's as the destination address.
3. But a broadcast IP datagram cannot pass through any router. To solve the problem, there is a need for an intermediary.
4. One of the hosts in local network (or a router that can be configured to operate at the application layer) can be used as a relay . The host in this case is called a **relay agent**.
5. The relay agent knows the unicast address of a BOOTP server. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the BOOTP server.
6. The packet, carrying a unicast destination address, is routed by any router and reaches the BOOTP server.
7. The BOOTP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent.
8. BOOTP server sends a BOOTP reply message to the relay agent.
9. The relay agent, after receiving the reply, sends it to the BOOTP client.

### 4.1.2. Limitations of BOOTP

- BOOTP is **not a dynamic configuration** protocol.
- BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator.

# DHCP

The **Dynamic Host Configuration Protocol** (DHCP) has been devised to

provide static anddynamic address allocation that can be manual or automatic as required.

- **Static Address Allocation** In this capacity DHCP acts as BOOTP does. It is backward compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.
- **Dynamic Address Allocation** DHCP has a second database with a pool of available IPaddresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

- When a DHCP client sends a DHCP request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned.
- On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.
- The dynamic aspect of DHCP is needed when a host moves from network to network oris connected and disconnected from a network (as is a subscriber to a s rvice provider).
- DHCP provides temporary IP addresses for a limited time. The addresses assigned from the pool are temporary addresses.
- The DHCP server issues a lease for a specific time. When the lease expires, the clientmust either stop using the IP address or renew the lease.
- The server has the option to agree or disagree with the renewal. If the server disagrees,the client stops using the address.

**DHCP Operation**

DHCP provides an automated way to distribute and update IP addresses and other configuration information on a network. A DHCP server provides this information to a DHCP client through the exchange of a series of messages, known as the DHCP conversation or the DHCP transaction displayed in Figure 21.13.

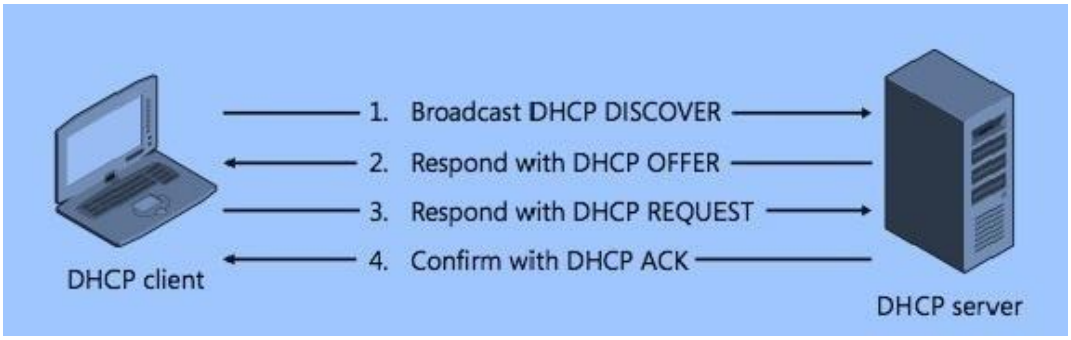



Figure 21.13:

DHCP OperationDHCP client goes through the

**f ur step** process:

**1.** A DHCP client sends a broadcast packet (**DHCP Discover**) to discover DHCP servers on the LAN segment.

**2.** The DHCP servers receive the **DHCP Discover** packet and respond with **DHCP Offer** packets, offering IP addressing information.

**3.** If the client receives the **DHCP Offer** packets from multiple DHCP servers, the first **DHCP Offer** packet is accepted. The client responds by broadcasting a **DHCP Request** packet, requesting network parameters from a single server.

**4.** The DHCP server approves the lease with a **DHCP Acknowledgement (DHCP Ack)** packet. The packet includes the lease duration and other configuration information.

# 1. ICMP

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

- The IP protocol has no error-reporting or error-correcting mechanism.
- What happens if something goes wrong?
- What happens if a router must discard a datagram because it cannot find a router to thefinal destination, or because the time-to-live field has a zero value?
- What happens if the final destination host must discard all fragments of a datagrambecause it has not received all fragments within a predetermined time limit?

These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

- The IP protocol also lacks a mechanism for host and management queries.
- A host sometimes needs to determine if a router or another host is alive.
- And sometimes a network administrator needs information from another host or router.

**The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol.**

## 1.1. Types of Messages

ICMP messages are divided into two broad categories: **Error-reporting messages and Querymessages**

The **error-reporting messages** report problems that a router or a host (destination) mayencounter when it processes an IP packet.

The **query messages**, which occur in pairs, help a host or a network manager get specificinformation from a router or another host.

## 1.2. Message Format

An ICMP message has an **8-byte header** and a **variable-size data section**. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 21.14 shows:

The first field, **ICMP type**, defines the type of the message.

The **code field** specifies the reason for the particular message type.

The last common field is the **checksum field** used for

securing ICMP header.The rest of the header is specific for

each message type.

The **data section** in error messages carries information for finding the original packetthat had the error.

In ICMP query messages, the data section carries extra information based on the type ofthe query.

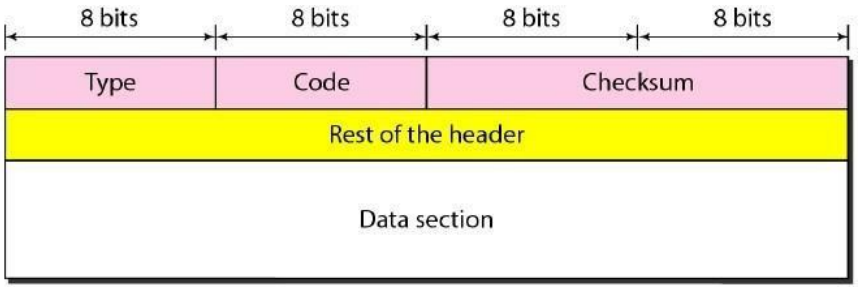



Figure 21.14 General format of ICMP messages

### 1.3. ICMP Encapsulation:

ICMP itself is a network layer protocol. However it's messages are not passed directly to datalink layer. Instead the messages are first encapsulated inside IP datagrams before going to the lower layer (see Figure 21.15).

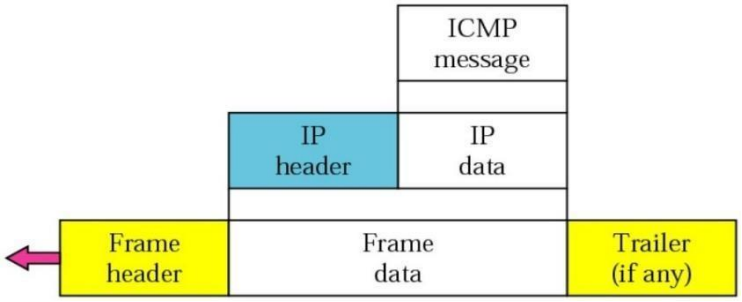



Figure 21.15 Contents of data field for the error messages

### 1.4. Error Reporting Messages

One of the main responsibilities of ICMP is to report errors.

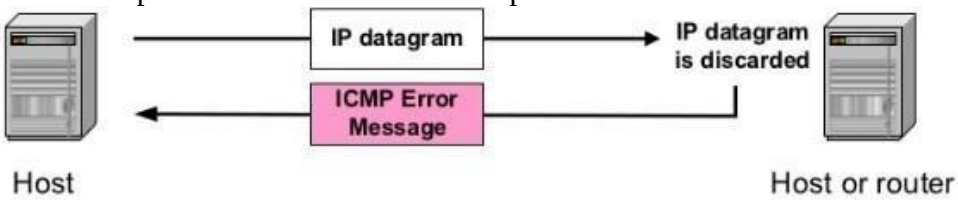



Figure 21.16 ICMP Error Reporting Message

Error messages are typically sent when a datagram is discarded due to some error asdisplayed in Figure 12.16.

Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.

**Five types of errors** are handled: *destination unreachable, source quench, timeexceeded,parameter problems, and redirection* (see Figure 21.17).

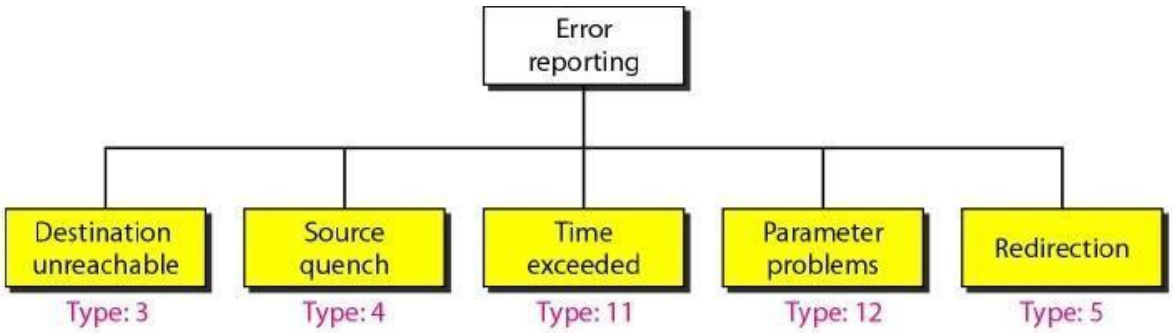



Figure 21.17 Error-reporting messages

**a. Destination Unreachable**

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

**b. Source Quench**

- **The source-quench message in ICMP was designed to add a kind of flow control to the IP.**
- When a router or host discards a datagram due to congestion**, it sends a source- quench message to the sender of the datagram**. This message has **two purposes**.
- **First**, it informs the source that the datagram has been discarded.
- **Second**, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process.

**c. Time Exceeded**

The time-exceeded message is generated in **two cases**:

**Case1**: As routers use routing tables to find the next hop (next router) that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loopor a cycle, going from one router to the next or visiting a series of routers endlessly. Each datagram contains a field called *time to live* that controls this situation. When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source.

**Case2**: A time-exceeded message is also generated when not all fragments that make upa message arrive at the destination host within a certain time limit.

**d. Parameter Problem**

Any **ambiguity in the header** part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.

**e. Redirection**

- This concept of redirection is shown in Figure 21.18. Host A wants to send a datagram to host B.

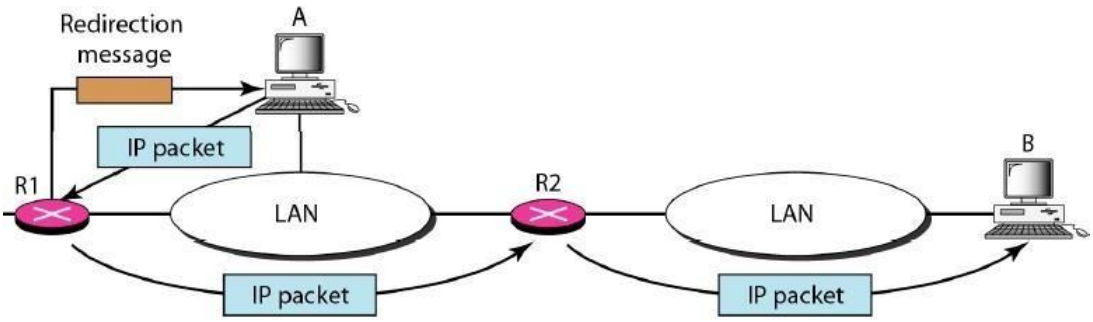



Figure 21.18 Redirection concept

- Router R2 is obviously the most efficient routing choice, but host A did notchoose router R2. The datagram goes to R1 instead.
- Router R1, after consulting its table, finds that the packet should have gone toR2.
- It sends the packet to R2 and, at the same time, sends a redirection message tohost A.
- Host A's routing table can now be updated.

### 1.5. ICMP Query Messages

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of **four different pairs of messages**, as shown in Figure 21.19.

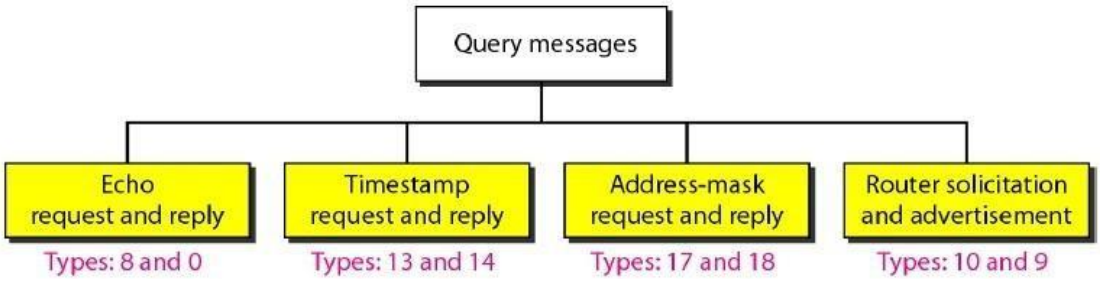



Figure 21.19 Query messages

In this type of ICMP message, a node sends a ICMP request message that is answered ina specific format as ICMP reply by the destination node, depicted in Figure 21.20.

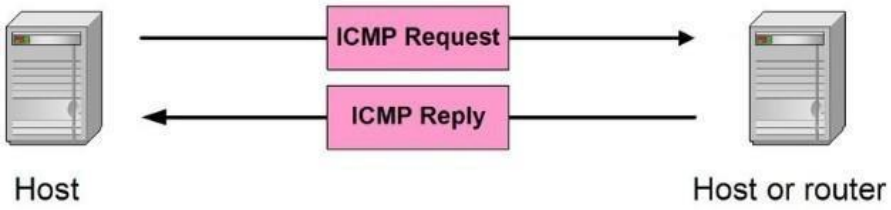



Figure 21.20 ICMP Query Message

A query message is encapsulated in an IP packet, which in turn is encapsulated in a data link layer frame.

However, in this case, no bytes of the original IP are included in the message, as shownin Figure 21.21.

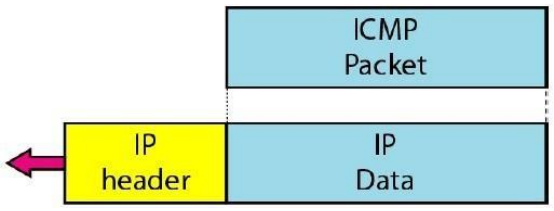



Figure 21.21 Encapsulation of ICMP query messages

**a. Echo Request and Echo Reply**

The echo-request and echo-reply messages are designed for diagnostic purposes. The combination of echo-request and echo-reply messages determines whether two systems (hosts or routers) can communicate with each other Figure 21.22. It also confirms that the intermediate routers are receiving, processing, and forwarding IP datagrams.

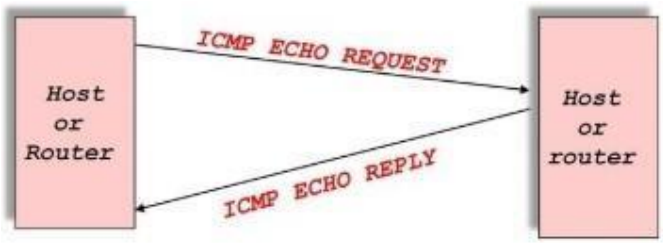



Figure 21.22 ICMP Echo Request and Echo Reply

Today, most systems provide a version of the ***ping* command** that can create a series (instead of just one) of echo-request and echo-reply messages, providing statistical information. We can use the *ping* program to find if a host is alive and responding.

**b. Timestamp Request and Timestamp Reply**

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

**c. Address-Mask Request and Address-Mask Reply**

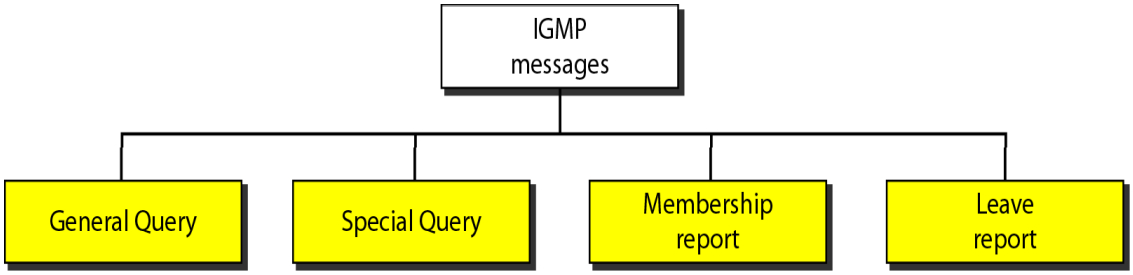
A host may know its IP address, but it may not know the corresponding mask. For example, a host may know its IP address as 159.31.17.24, but it may not know that the corresponding mask is /24. To obtain its mask, a host sends an address-mask-request message to a router on the LAN. If the host knows the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

**d. Router Solicitation and Router Advertisement**

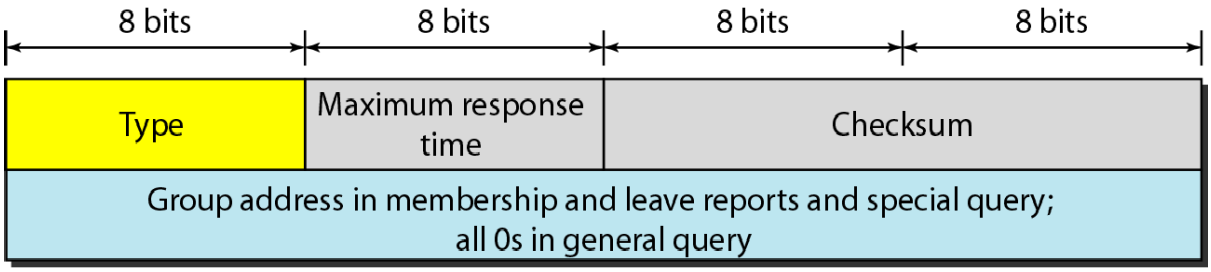
The router-solicitation and router-advertisement messages can help a host to check whether the neighboring routers are alive and functioning. A host can broadcast (or multicast) a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited.

## IGMP:

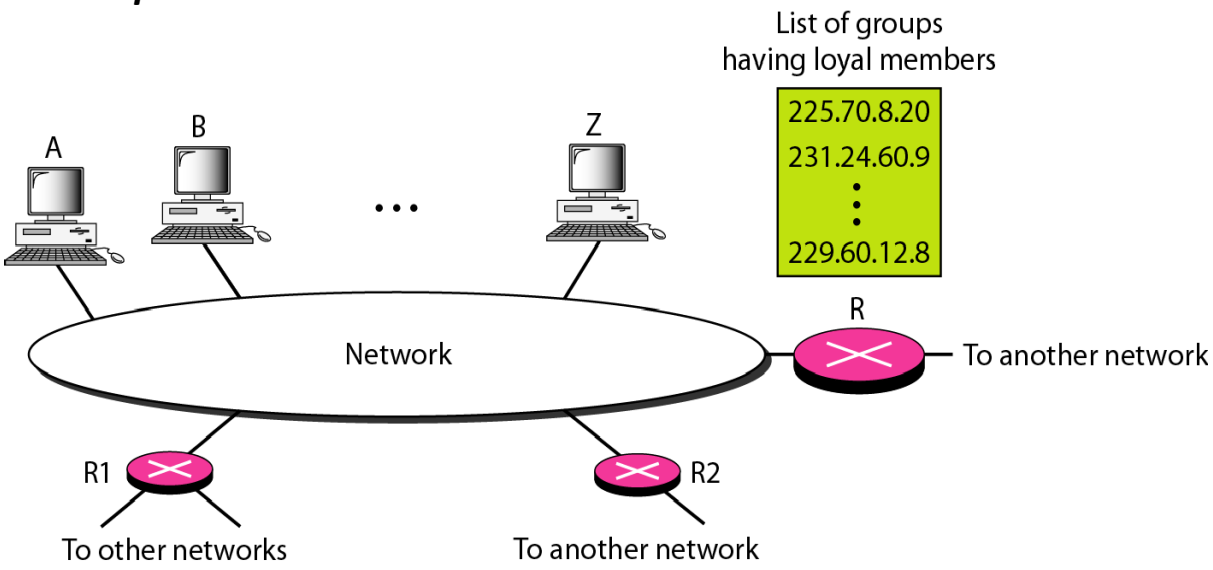
The IP protocol can be involved in two types of communication: unicasting and multicasting. The Internet Group Management Protocol (IGMP) is one of the necessary, but not sufficient, protocols that is involved in multicasting.. IGMP is a companion to the IP protocol.




- The Internet Group Management Protocol (IGMP) is a protocol that allows several devices to share one IP address so they can all receive the same data.
- IGMP is a network layer protocol used to set up multicasting on networks that use the Internet Protocol version 4 (IPv4).
- Specifically, IGMP allows devices to join a multicasting group.
- Multicasting is when a group of devices all receive the same messages or packets.
- Multicasting works by sharing an IP address between multiple devices. Any network traffic directed at that IP address will reach all devices that share the IP address, instead of just one device.
- This is much like when a group of employees all receive company emails directed at a certain email alias.
- Membership reports: Devices send these to a multicast router in order to become a member of a multicast group.
- "Leave group" messages: These messages go from a device to a router and allow devices to leave a multicast group.
- General membership queries: A multicast-capable router sends out these messages to the entire connected network of devices to update multicast group membership for all groups on the network.
- Group-specific membership queries: Routers send these messages to a specific multicast group, instead of the entire network.



***IGMP operation:***



# Distance Vector Routing

- In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. The term **Vector** means a **Table**.
- In this protocol each node maintains a table of minimum distances to every node.
- The table at each node also guides the packets to the desired node by showing the next hop in the route.
- The table for node A shows how we can reach to any node from node A.
- Ex: From node A the least cost to reach node E is 6. The route passes through C.

| To | Cost | Next |
|---|---|---|
| **A** | 0 | - |
| **B** | 5 | - |
| **C** | 2 | - |
| **D** | 3 | - |
| **E** | 6 | C |

- The Distance Vector Routing protocol follows these basic steps:

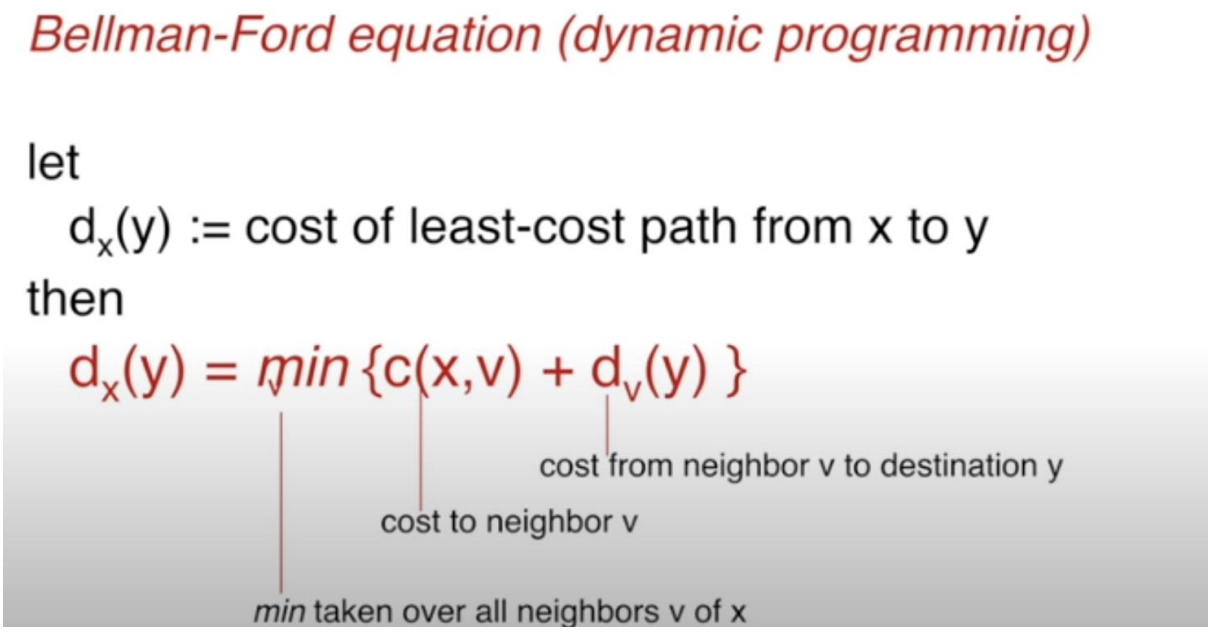
**1. Initialization:** Each router in the network is configured with its own distance vector table, which lists the distance (in terms of hop count or other metrics) to every other network in the inter-network.

**2. Sending distance vectors:** Each router sends its distance vector table to its immediate neighbors.(Sharing)

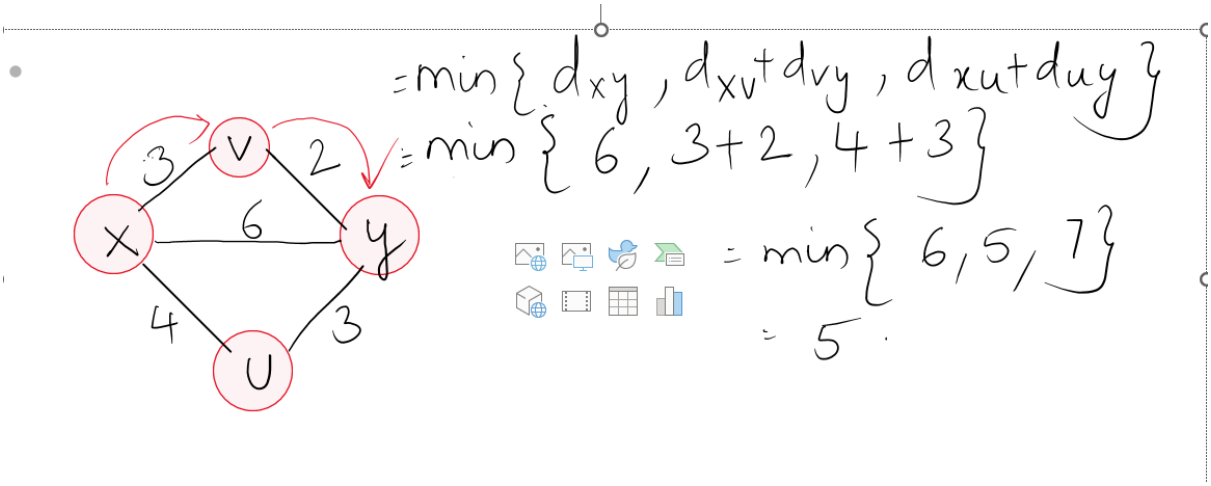
**3. Updating distance vectors:** Upon receiving a distance vector from a neighbor, a router updates its own table by comparing the distance to a network via its neighbor with the distance currently listed in its own table. If the new distance is shorter, the router updates its table with the new information.

**4. Calculating best path:** Using the information in its distance vector table, each router calculates the best path to each network and updates its routing table accordingly.

- **5. Periodic updates:** The sending and updating distance vectors are repeated periodically, typically every 30 seconds. This allows the network to adapt to changes in topology quickly.
- **6. Convergence:** The process of sending and updating distance vectors continues until all routers in the network have the most up-to-date information and have converged on the best path to each network.

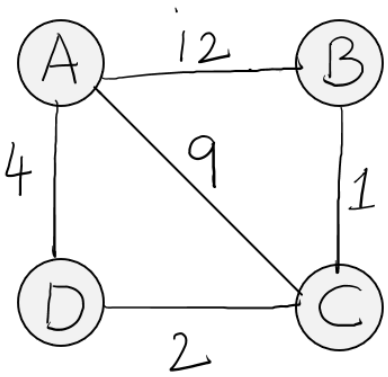



Example:




Example:

1. Initialization




A's table

| To | Cost | Next |
|---|---|---|
| A | 0 | A |
| B | 12 | B |
| C | 9 | C |
| D | 4 | D |

B's table

| To | Cost | Next |
|---|---|---|
| A | 12 | A |
| B | 0 | B |
| C | 1 | C |
| D | ∞ | - |

C's table

| To | Cost | Next |
|---|---|---|
| A | 9 | A |
| B | 1 | B |
| C | 0 | C |
| D | 2 | D |

D's table

| To | Cost | Next |
|---|---|---|
| A | 4 | A |
| B | ∞ | - |
| C | 2 | C |
| D | 0 | D |

2. Updation:
   A receives distance vectors from B, C, and D.
   It updates its table using their neighbors table.
   A reaches B initially with a cost of 12.
   After getting the neighbors tables A updates its table using Bellmans Ford Equation.
   Example:

A

| To | Cost | Next |
|---|---|---|
| A | 0 | A |
| B | 10 | C |
| C | 6 | D |
| D | 4 | D |

B

| To | Cost | Next |
|---|---|---|
| A | 10 | C |
| B | 0 | B |
| C | 1 | C |
| D | 3 | C |

C

| To | Cost | Next |
|---|---|---|
| A | 6 | D |
| B | 1 | B |
| C | 0 | C |
| D | 2 | D |

D

| To | Cost | Next |
|---|---|---|
| A | 4 | A |
| B | 3 | C |
| C | 2 | C |
| D | 0 | D |

All routers use the same process to update their tables.

3. Final Solution

A

| To | Cost | Next |
|---|---|---|
| A | 0 | A |
| B | 7 | D,C |
| C | 6 | D |
| D | 4 | D |

B

| To | Cost | Next |
|---|---|---|
| A | 7 | C,D |
| B | 0 | B |
| C | 1 | C |
| D | 3 | C |

C

| To | Cost | Next |
|---|---|---|
| A | 6 | D |
| B | 1 | B |
| C | 0 | C |
| D | 2 | D |

D

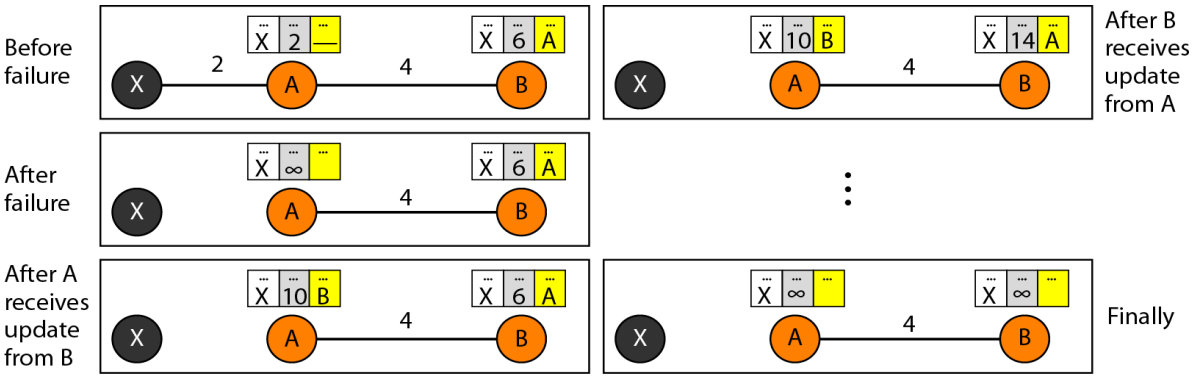
| To | Cost | Next |
|---|---|---|
| A | 4 | A |
| B | 3 | C |
| C | 2 | C |
| D | 0 | D |

Count to Infinity Problems Two-Node Loop Instability

A problem with distance vector routing is instability, which means that a network using this protocol can become unstable.
Consider the above figure:
It shows a system with 3 nodes: X, A and B.

- 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 there will be no problem because B can identify by looking at the table value ∞.




**Problem is:** The system becomes unstable if B sends its routing table to A before receiving A's routing table.

- Node A receives the update and, assuming that B has found a way to reach X and immediately updates its routing table.
- Based on the triggered update strategy, A sends its new update to B.
- Now B thinks that something has been changed around A and updates its routing table.
- The cost of reaching X increases gradually until it reaches infinity.
- At this moment, both A and B know that X cannot be reached.

During this counting to infinity the system is not stable:

- Node A thinks that the route to X is via B.

- Node B thinks that the route to X is via A.
- If A receives a packet destined for X, it goes to B and then comes back to A.
- 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.

## Link State Routing (LSR):

In Link State Routing, Each node in the domain has the entire topology of the domain –

- List of nodes and links
- How they are connected including the type
- Cost (metric)
- Condition of the links (up or down)

**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 (Distribution) of LSPs to every other router called **Flooding**. The flooding can be done in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

**Creation of Link State Packet (LSP)**

A link state packet can carry a large amount of information such as the node identity, the list of links, a sequence number, and age etc.

- Node identity and the List of links are needed to make the topology.
- Sequence number facilitates flooding and distinguishes new LSPs from old ones.
- Age prevents old LSP's from remaining LSP's in the domain for a longtime.

**Flooding of LSP's**

After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding.

Flooding will be done based on the following:

1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have.

Each and every LSP will be given a Sequence number at the time of their creation. Comparison of sequence numbers determines which LSP is older and which LSP is latest. If the newly arrived LSP is older than the one it already has, then the node discards the LSP.

## Formation of Shortest Path Tree: Dijkstra Algorithm

- After receiving all LSPs, each node will have a copy of the whole topology.
- The topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.
- A tree is a graph of nodes and links, where one node is called Root.
- A shortest path tree is a tree in which the path between the root and every other node is the shortest.
- The Dijkstra's algorithm creates a shortest path tree from a graph.

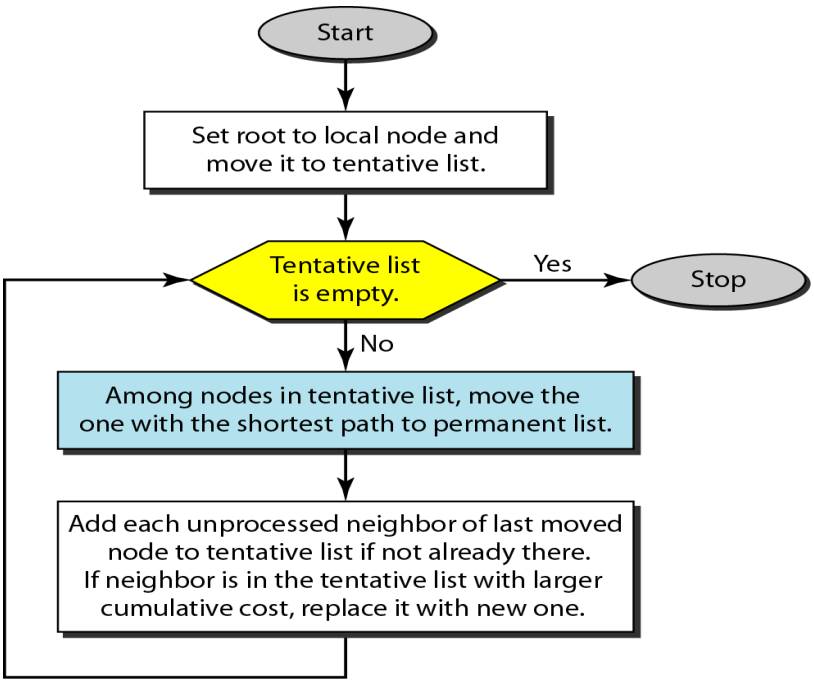
The algorithm divides the nodes into two sets:

1. Tentative nodes

2. Permanent nodes

Dijkstra's algorithm finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.

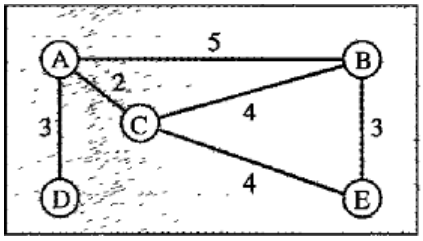
**Flow Chart of Dijksta's Algorithm**




**Example:**

Consider the below graph with five nodes: A,B,C,D,E.

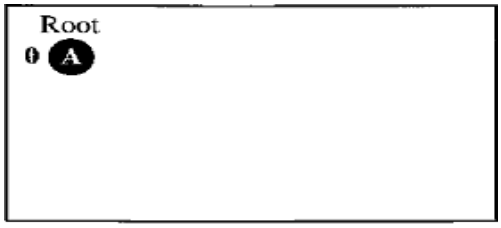
- Apply the Dijkstra's algorithm to node A.
- To find the shortest path in each step, we need the cumulative cost from the root to each node, which is shown next to the node.




Topology

At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.

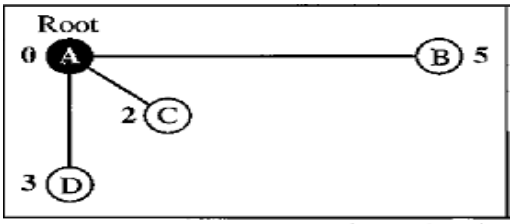
**Step 1**: We make node A the root of the tree and move it to the tentative list. Our two lists are Permanent list: **Empty** Tentative list: **A(0)**




1. Set root to A and move A to tentative list.

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

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



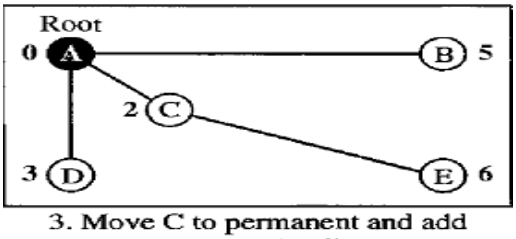

2. Move A to permanent list and add B, C, and D to tentative list.

C(2), D(3)

**Step 3:** Node C has the shortest cumulative cost from all nodes in the tentative list.

- We move C to the permanent list.
- Node C has three neighbors, but node A is already processed, which makes the unprocessed neighbors just B and E.
- However, B is already in the tentative list with a cumulative cost of 5.
- Node A could also reach node B through C with a cumulative cost of 6.
- Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it.

Our new lists are: Permanent list: A(0),C(2) Tentative list: B(5),



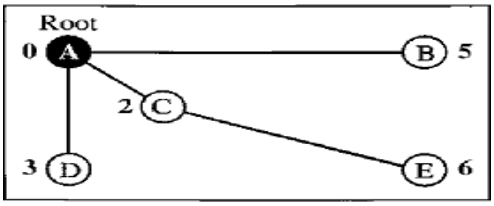

3. Move C to permanent and add E to tentative list.

D(3),E(6).

**Step 4:** Node D has the shortest cumulative cost of all the nodes in the tentative list.

- We move D to the permanent list. Node D has no unprocessed neighbor to be added to the tentative list.

Our new lists are: Permanent List: A(0),C(2),D(3) Tentative List: B(5),E(6).

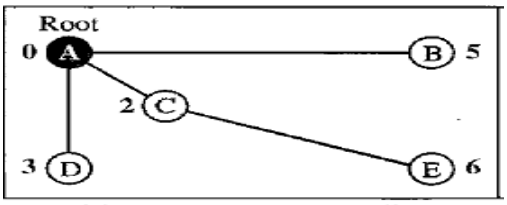



4. Move D to permanent list.

**Step 5:** Node B has the shortest cumulative cost of all the nodes in the tentative list.

- We move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (i.e. just node E).
- E(6) is already in the list with a smaller cumulative cost.
- The cumulative cost to node E, as the neighbor of B, is 8. We keep node E(6) in the tentative list.

Our new lists are: Permanent list: A(0), B(5),C(2),D(3) Tentative list: E(6)

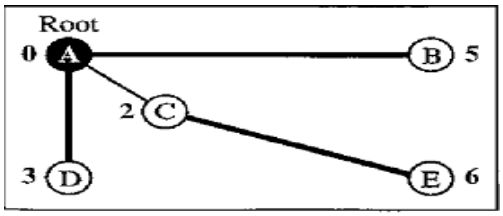



5. Move B to permanent list.

**Step 6:** Node E has the shortest cumulative cost from all nodes in the tentative list.

- Move E to the permanent list. Node E has no neighbor. Now the tentative list is empty.
- We stop the process here. The shortest path tree is ready for graph ABCDE.

The finalists are: Permanent list: A(0), B(5), C(2), D(3), E(6) Tentative list: Empty




6. Move E to permanent list (tentative list is empty).

**Calculation of Routing Table from Shortest Path Tree**

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

from the root. The below table shows routing table for Node A.

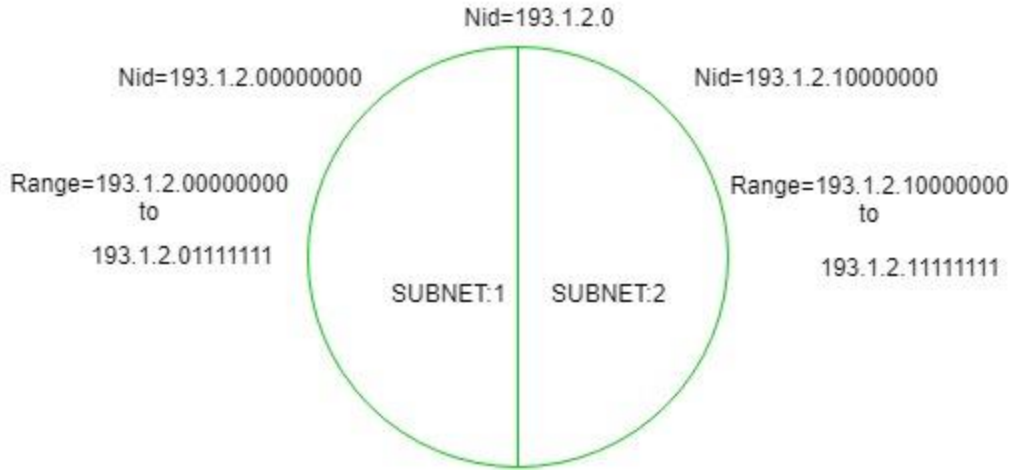
| Node | Cost | Next Router |
|---|---|---|
| A | 0 | - |
| B | 5 | - |
| C | 2 | - |
| D | 3 | - |
| E | 6 | C |

When a bigger network is divided into smaller networks, to maintain security, then that is known as Subnetting. So, maintenance is easier for smaller networks. For example, if we consider a class A address, the possible number of hosts is $2^{24}$ for each network, it is obvious that it is difficult to maintain such a huge number of hosts, but it would be quite easier to maintain if we divide the network into small parts.

## Uses of Subnetting

1. Subnetting helps in organizing the network in an efficient way which helps in expanding the technology for large firms and companies.
2. Subnetting is used for specific staffing structures to reduce traffic and maintain order and efficiency.
3. Subnetting divides domains of the broadcast so that traffic is routed efficiently, which helps in improving network performance.
4. Subnetting is used in increasing network security.

The network can be divided into two parts: To divide a network into two parts, you need to choose one bit for each Subnet from the host ID part.




In class C the first 3 octets are network bits so it remains as it is.

- **For Subnet-1:** The first bit which is chosen from the host id part is zero and the range will be from (193.1.2.00000000 till you get all 1's in the host ID part i.e, 193.1.2.01111111) except for the first bit which is chosen zero for subnet id part.

Thus, the range of subnet 1 is: **193.1.2.0 to 193.1.2.127**

```
Subnet id of Subnet-1 is : 193.1.2.0
The direct Broadcast id of Subnet-1 is: 193.1.2.127
The total number of hosts possible is: 126 (Out of 128,
2 id's are used for Subnet id & Direct Broadcast id)
The subnet mask of Subnet- 1 is: 255.255.255.128
```

- **For Subnet-2:** The first bit chosen from the host id part is one and the range will be from (193.1.2.100000000 till you get all 1's in the host ID part i.e, 193.1.2.11111111).

Thus, the range of subnet-2 is: **193.1.2.128 to 193.1.2.255**

```
Subnet id of Subnet-2 is : 193.1.2.128
The direct Broadcast id of Subnet-2 is: 193.1.2.255
The total number of hosts possible is: 126 (Out of 128,
2 id's are used for Subnet id &  Direct Broadcast id)
The subnet mask of Subnet- 2 is: 255.255.255.128
The best way to find out the subnet mask of a subnet
is to set the fixed bit of host-id to 1 and the rest to 0.
```

Finally, after using the subnetting the total number of usable hosts is reduced from 254 to 252.

**Note:**

1. To divide a network into four ($2_2$) parts you need to choose two bits from the host id part for each subnet i.e, (00, 01, 10, 11).
2. To divide a network into eight ($2_3$) parts you need to choose three bits from the host id part for each subnet i.e, (000, 001, 010, 011, 100, 101, 110, 111) and so on.
3. We can say that if the total number of subnets in a network increases the total number of usable hosts decreases.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function to convert an IP address to a 32-bit integer
unsigned int ipToInt(char* ip) {
   unsigned int a, b, c, d;
   sscanf(ip, "%u.%u.%u.%u", &a, &b, &c, &d);
   return (a << 24) | (b << 16) | (c << 8) | d;
}

// Function to convert a 32-bit integer to an IP address
void intToIp(unsigned int ip, char* buffer) {
   sprintf(buffer, "%u.%u.%u.%u", (ip >> 24) & 0xFF, (ip >> 16) & 0xFF, (ip >> 8) &
0xFF, ip & 0xFF);
}

// Function to calculate the subnet mask from a prefix length
unsigned int calculateSubnetMask(int prefixLength) {
   return prefixLength == 0 ? 0 : ~((1 << (32 - prefixLength)) - 1);
}

int main() {
   char ip[16];
```

```
    int prefixLength, newPrefixLength;
    unsigned int subnetMask, newSubnetMask, ipInt;
    char buffer[16];

    // Input IP address and prefix length
    printf("Enter IP address (e.g., 192.168.1.0): ");
    scanf("%s", ip);
    printf("Enter current prefix length (e.g., 24): ");
    scanf("%d", &prefixLength);

    // New prefix length for creating two subnets
    newPrefixLength = prefixLength + 1;

    // Convert IP address to integer
    ipInt = ipToInt(ip);

    // Calculate original subnet mask and new subnet mask
    subnetMask = calculateSubnetMask(prefixLength);
    newSubnetMask = calculateSubnetMask(newPrefixLength);

    // Calculate the number of hosts per subnet
    int hostsPerSubnet = (1 << (32 - newPrefixLength)) - 2; // subtract 2 for
network and broadcast addresses

    printf("\nNumber of subnets: 2\n");
    printf("Number of hosts per subnet: %d\n", hostsPerSubnet);

    // Generate subnets
    for (int i = 0; i < 2; i++) {
        unsigned int subnetNetwork = (ipInt & subnetMask) | (i << (32 -
newPrefixLength));
        unsigned int subnetBroadcast = subnetNetwork | ~newSubnetMask;
        unsigned int firstHost = subnetNetwork + 1;
        unsigned int lastHost = subnetBroadcast - 1;

        printf("\nSubnet %d:\n", i + 1);
        printf("Network Address: ");
        intToIp(subnetNetwork, buffer);
        printf("%s\n", buffer);

        printf("Broadcast Address: ");
        intToIp(subnetBroadcast, buffer);
```

```
        printf("%s\n", buffer);

        printf("Subnet Mask: ");
        intToIp(newSubnetMask, buffer);
        printf("%s\n", buffer);

        printf("First Host: ");
        intToIp(firstHost, buffer);
        printf("%s\n", buffer);

        printf("Last Host: ");
        intToIp(lastHost, buffer);
        printf("%s\n", buffer);
    }

    return 0;
}
```

INPUT:

```
Enter IP address (e.g., 192.168.1.0): 193.1.2.0
Enter current prefix length (e.g., 24): 24
```

OUTPUT:

```
Number of subnets: 2
Number of hosts per subnet: 126

Subnet 1:
Network Address: 193.1.2.0
Broadcast Address: 193.1.2.127
Subnet Mask: 255.255.255.128
First Host: 193.1.2.1
Last Host: 193.1.2.126

Subnet 2:
Network Address: 193.1.2.128
Broadcast Address: 193.1.2.255
Subnet Mask: 255.255.255.128
First Host: 193.1.2.129
Last Host: 193.1.2.254

--------------------------------
```

LAB EXPERIMENT -4

## AIM: Write a C program to Implement distance vector routing algorithm for obtaining routing tables at each node.

## Distance Vector Routing:

- **Nodes and Distance Vectors**: The program initializes a set of nodes and their distance vectors.
- **Bellman-Ford Algorithm**: This is used to update the distance vectors until they stabilize.
- **Routing Table Construction**: Each node maintains a routing table with the next hop and the distance to each destination.

## Explanation:

1. **Initialization**:
   - The `initialize` function sets up the distance vectors and routing tables for each node.
   - If there's a direct link between two nodes, the initial distance is set to the cost of that link, and the next hop is set to the destination node itself. Otherwise, the distance is set to `INF` (representing no direct link), and the next hop is set to `-1`.
2. **Distance Vector Routing Algorithm**:
   - The `distanceVectorRouting` function uses the Bellman-Ford algorithm to update the distance vectors and next hops.
   - The algorithm iteratively updates the distance vectors until no further updates are necessary, indicating convergence.
3. **Printing the Routing Tables**:
   - The `printRoutingTable` function displays the routing table for each node, showing the destination, next hop, and distance.
4. **Main Function**:
   - The `main` function reads the number of nodes and the cost matrix from the user, initializes the data structures, runs the distance vector routing algorithm, and prints the routing tables.

CODE:

```
#include <stdio.h>

#include <stdlib.h>



#define INF 9999

#define MAX_NODES 10
```

```
// Function to initialize distance vector and routing table
void initialize(int numNodes, int costMatrix[MAX_NODES][MAX_NODES], int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            distVector[i][j] = costMatrix[i][j];
            if (costMatrix[i][j] != INF && i != j) {
                nextHop[i][j] = j;
            } else {
                nextHop[i][j] = -1;
            }
        }
    }
}

// Function to print routing table for each node
void printRoutingTable(int numNodes, int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    for (int i = 0; i < numNodes; i++) {
        printf("Routing table for node %d:\n", i);
        printf("Destination\tNext Hop\tDistance\n");
        for (int j = 0; j < numNodes; j++) {
            if (distVector[i][j] == INF) {
                printf("%d\t\t-\t\tINF\n", j);
            } else {
                printf("%d\t\t%d\t\t%d\n", j, nextHop[i][j], distVector[i][j]);
            }
        }
```

```
        printf("\n");
    }
}

// Function to implement Distance Vector Routing algorithm
void distanceVectorRouting(int numNodes, int costMatrix[MAX_NODES][MAX_NODES], int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    int updated;

    do {
        updated = 0;
        for (int i = 0; i < numNodes; i++) {
            for (int j = 0; j < numNodes; j++) {
                for (int k = 0; k < numNodes; k++) {
                    if (distVector[i][k] + distVector[k][j] < distVector[i][j]) {
                        distVector[i][j] = distVector[i][k] + distVector[k][j];
                        nextHop[i][j] = nextHop[i][k];
                        updated = 1;
                    }
                }
            }
        }
    } while (updated);
}

int main() {
    int numNodes, costMatrix[MAX_NODES][MAX_NODES];
    int distVector[MAX_NODES][MAX_NODES];
```

```
    int nextHop[MAX_NODES][MAX_NODES];

    printf("Enter the number of nodes: ");
    scanf("%d", &numNodes);

    printf("Enter the cost matrix (use %d for INF):\n", INF);
    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            scanf("%d", &costMatrix[i][j]);
        }
    }

    initialize(numNodes, costMatrix, distVector, nextHop);
    distanceVectorRouting(numNodes, costMatrix, distVector, nextHop);
    printRoutingTable(numNodes, distVector, nextHop);

    return 0;
}
```

INPUT:

```
Enter the number of nodes: 4
Enter the cost matrix (use 9999 for INF):
0       12      9       4
12      0       1       9999
9       1       0       2
4       9999    2       0
```

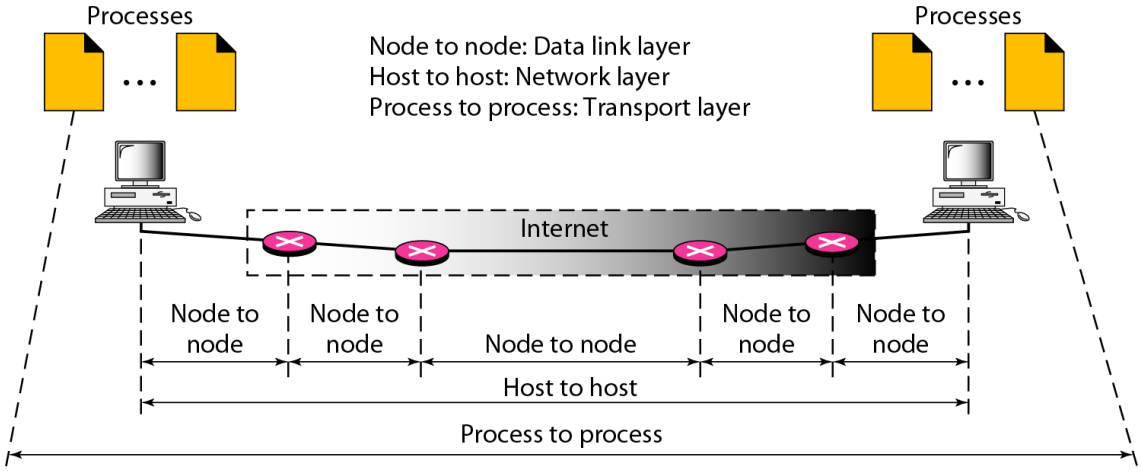
CHECK THE OUTPUT FOR THE ABOVE INPUT.

## PROCESS TO PROCESS DELIVERY

A Process is an Application program that runs on the host. At any moment several processes may be running on the source host and destination host.

Transport Layer is responsible for delivery of a packet from one process on source host to another process on destination host.

Two processes communicate in a client/server relationship.




### Client/Server Paradigm

A Client is a process on a local host, whereas Server is a process on remote host. A Client needs services from Server. Both Client and Server processes have the same name.

**Example:** To get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

For communication to be done between two processes we must have to define following:

1. Local host
2. Local process
3. Remote host
4. Remote process

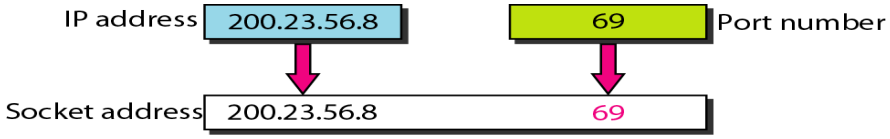
### Addressing

- Transport layer needs an address called a **Port Number** or **Port Address** to choose among multiple processes running on the destination host.
- The port numbers are 16-bit integers between 0 and 65,535.
- Destination Port number is needed for Delivery. Source Port number is needed for the Reply.
- At the client side the port numbers are defined randomly whereas at the server side it uses Well-Known Port numbers.

### Socket address

- The combination of an IP address and a port number is called a **Socket address**. UDP or TCP header contains the Port numbers.

- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.




## TRANSPORT LAYER PROTOCOL

There are 3 transport layer protocols are implemented:

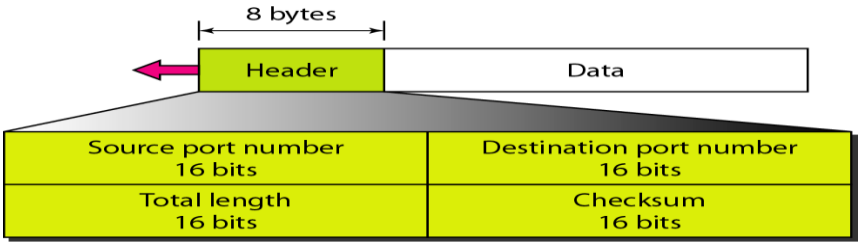
1. UDP
2. TCP
3. SCTP

## USER DATAGRAM PROTOCOL (UDP)

- UDP is called a connectionless, unreliable transport protocol.
- UDP is a very simple protocol using a minimum of overhead. UDP performs very limited error checking.
- UDP is used when a process wants to send a small message and does not need reliability.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

### User Datagram

UDP packets are called User Datagrams. Datagrams have fixed size header of length 8




bytes.

Fields of user datagram are:

**Source port number (16 bits)**

- This is the port number used by the process running on the source host.

**Destination port number**

- This is the port number used by the process running on the destination host.

Note: Source and Destination port number can range from 0 to 65,535.

**Total Length (16 bits)**

The total length includes UDP header plus Data, total length ranges from 0-65535.

**Checksum (16 bits)**

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

**UDP Operation**

UDP uses the following four concepts:

1. Connectionless Service

2. No Flow and Error control
3. Encapsulation and Decapsulation
4. Queuing

**Connectionless Services**

- UDP provides a connectionless service. There is no connection establishment and no connection termination. Each user datagram sent by UDP is an independent datagram.
- The user datagrams are not numbered. Each user datagram can travel on a different path.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- UDP processes are capable of sending short messages only.

**No Flow and Error Control**

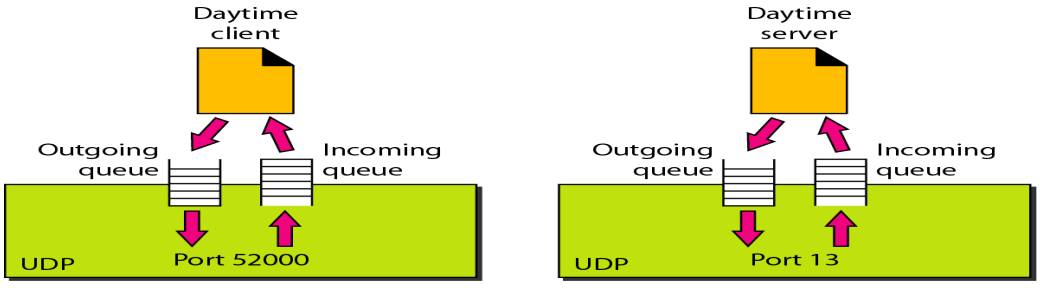
- There is no error control mechanism in UDP except for the checksum.

**Encapsulation and Decapsulation**

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

**Queuing**

- Queues are associated with ports in UDP. Queues are associated with each process.
- Queues are created when a process is started. There are two types of queues are created: Incoming queue and Outgoing queue.
- A process wants to communicate with multiple processes; it obtains only one port number and one outgoing queue and one incoming queue.
- The queues function as long as the process is running. When the process terminates, the queues aredestroyed.




**Uses of UDP**

The following lists some uses of the UDP protocol:

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow and error control mechanisms.
- UDP is a suitable transport protocol for multicasting.
- UDP is used for management processes such as SNMP.
- UDP is used for Route Updating Protocols such as Routing Information Protocol(RIP).

## TRANSMISSION CONTROL PROTOCOL (TCP)

TCP is called a connection-oriented, reliable, process-to-process transport protocol. It adds connection-oriented and reliability features to the services of IP.

### TCP Services

The following five services offered by TCP to the processes at the application layer

1. Process-to-Process Communication
2. Full-Duplex Communication
3. Connection-Oriented Service
4. Reliable Service
5. Stream Delivery Service

### Process-to-Process Communication

TCP provides process-to-process communication using Well-known port numbers.

### 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

- When a process at site A wants to send and receive data from another process at site B, the following steps will occur:
  1. The two TCPs establish a virtual 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 must be resent.
- TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

### Reliable Service

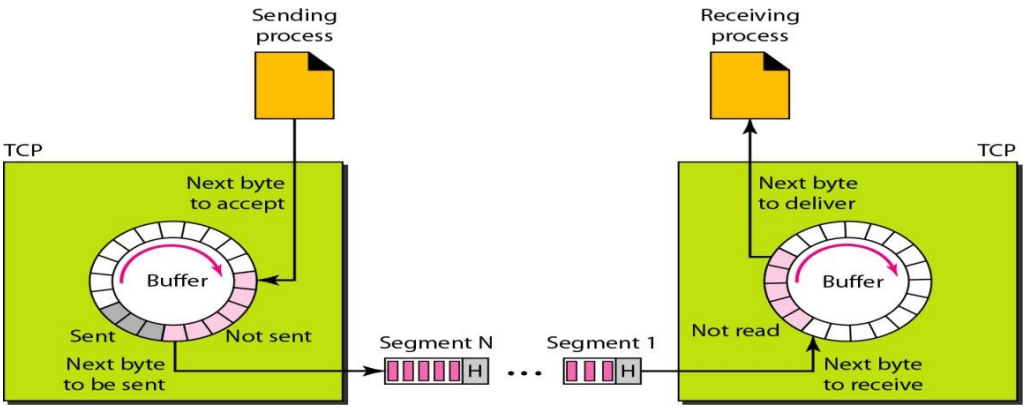
TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

### Stream Delivery Service

- TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
- The sending process produces the stream of bytes, and the receiving process consumes them.

### Sending and Receiving Buffers

- TCP needs buffers for storage, because the sending and the receiving processes may not write or read data at the same speed.
- The buffers are implemented by using Circular Array where each location carries1-byte.
- There are two types of buffers are implemented: **Sending buffer** and **Receiving buffer**.




- At the sending site TCP keeps bytes in the buffer that have been sent but not yet acknowledged until it receives an acknowledgment.
- After the bytes in the buffer locations are acknowledged, the locations are recycled and they are available for use by the sending process.
- At the receiver site the circular buffer is divided into two areas:
- One area contains empty chambers to be filled by bytes received from the network.
- Other are contain received bytes that can be read by the receiving process.
- When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

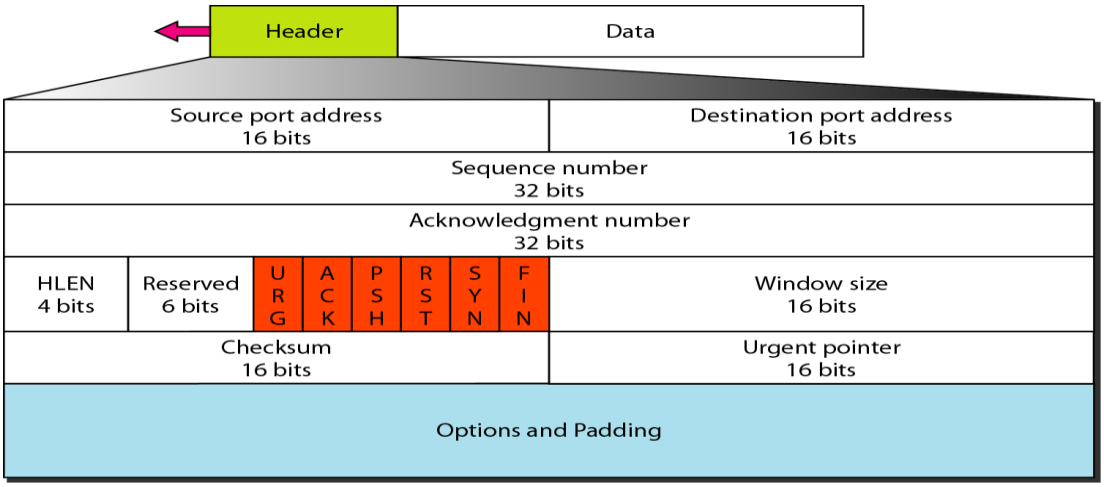
**Segments**

- TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment and delivers them to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- The segments are not necessarily the same size.

## TCP Segment

- A packet in TCP is called a segment.
- A segment consists of Segment Header and Data.
- The segment Header consists of 20-60 Byte. Header is 20 bytes if there are no options. If there are any options the length of the header varied upto 60bytes.

The segment format can be given as:




- **Source port address (16bit)**

It defines the port number of the application program in the host that is sending the segment.

- **Destination port address(16-bit)**

It defines the port number of the application program in the host that is receiving the segment.

- **Sequence number (32bit)**

This field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.

- **Acknowledgment number (32 bit)**

This field defines the byte number that the receiver of the segment is expecting to receive from the other party. Acknowledgment and data can be piggybacked together.

- **Header length (4bit)**

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

The value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).

- **Reserved (6 bit)**

This field is reserved for future use.

- **Window size (16bit)**

This field defines the size of the window in bytes that the other party must maintain. The maximum size of the window is 65,535 bytes.

This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.

- **Checksum (16bit)**

The checksum field in TCP is mandatory. For the TCP pseudo header, the value for the protocol field is 6.

- **Urgent pointer(l6-bit)**

This field is valid only if the urgent flag is set. It 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 (up to 40 bytes)** It defines the optional information in the TCP header.
- **Control flags (6bit)**

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

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

## TCP CONNECTION CONTROL:

Connection-oriented TCP transmission requires three phases:

1. Connection establishment
2. Data transfer
3. Connection termination

**Connection Establishment Phase**

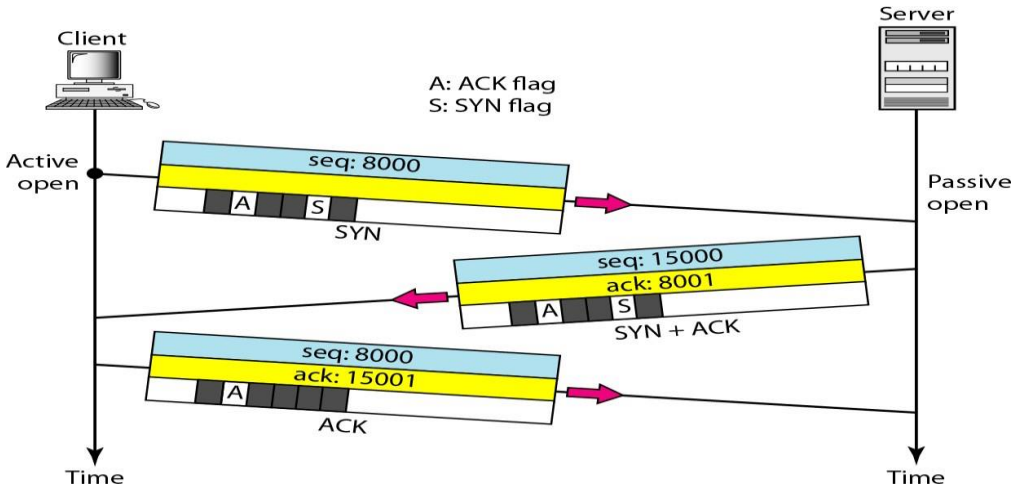
TCP uses Three way handshaking to establish a connection. The three way handshaking uses the sequence number, the acknowledgment number, the control flags and the window size.

- 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**.
- A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. The client program issues a request for an **Active open.**
- Now TCP starts the Three way handshaking protocol process.

The three steps in this phase are as follows:

1. The client sends the first segment a SYN segment. The SYN flag in control field is set. This segment is for synchronization of sequence numbers. The SYN segment does not carry real data. SYN consumes one sequence number. When the data transfer starts the sequence number is incremented by1.
2. The server sends the second segment a SYN + ACK segment with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

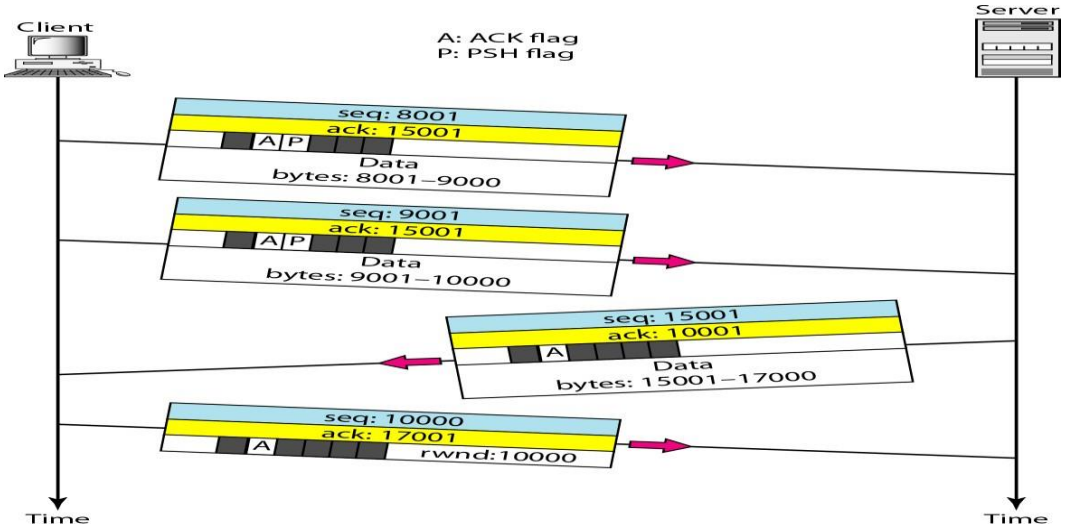
3. The client sends the third segment an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. The sequence number in this segment is the same as the one in the SYN segment.




**Data Transfer Phase**

- 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.

Consider the below figure that shows the data transfer after connection is established.

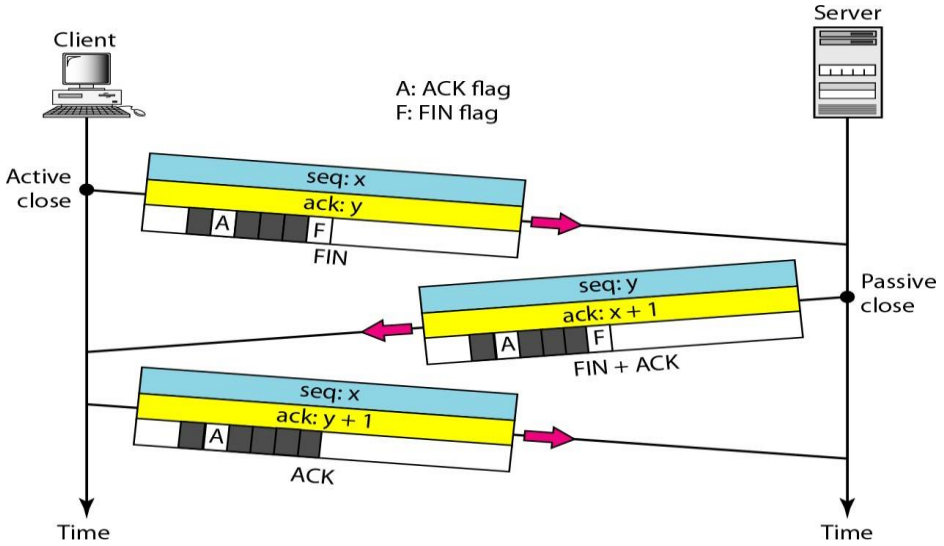



- The client sends 2000 bytes of data in two segments.
- The server then sends 2000 bytes in one segment.
- The client sends one more segment.
- The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.
- 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 does not set the push flag.

**Connection Termination**

Client and Server involved in exchanging data can close the connection is called Connection Termination. It is usually initiated by the client.

**Three way Handshaking** is also used to terminate the connection by following steps:




1. 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.

A FIN segment can include the last chunk of data sent by the client or it can be just a control segment. If it is only a control segment, it consumes only one sequence number.

2. The server TCP after receiving the FIN segment informs server process of the situation and sends the second segment a **FIN + ACK** segment to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

3. The TCP client 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 **(sequence number + 1)** received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

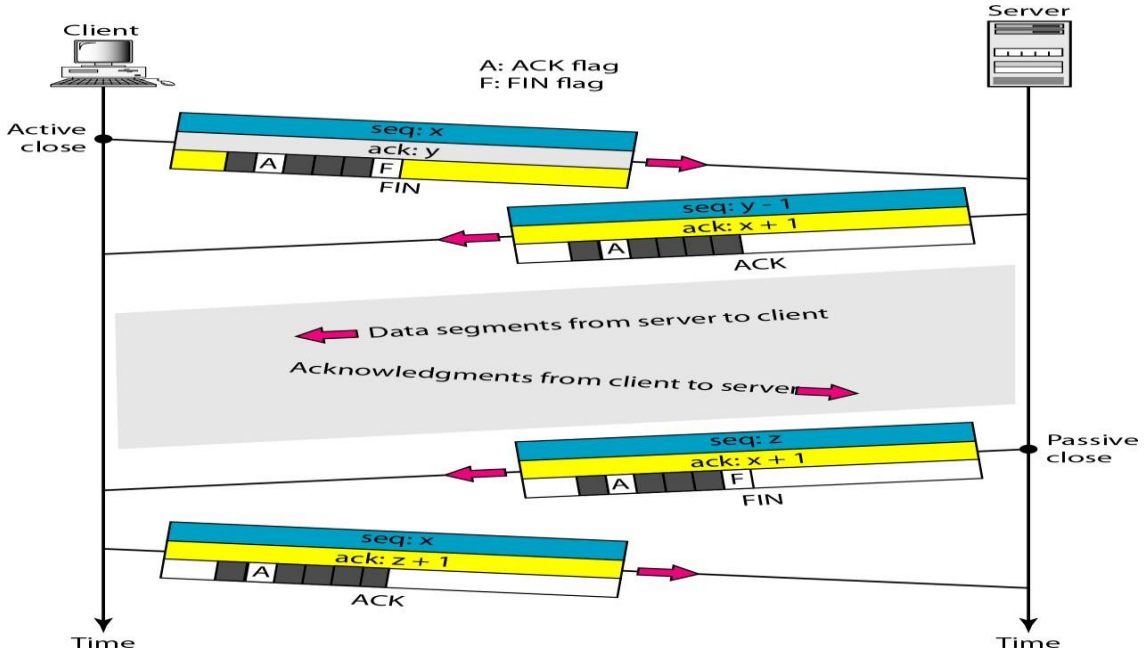
**Note:** In some rare situations Connection Termination can be done in Four way handshaking by using Half-close option.

**Four way Handshaking using Half-Close**

- In TCP one end can stop sending data while still receiving data is called a **Half-Close**. It is normally initiated by the client.
- It can occur when the server needs all the data before processing can begin.

Example: 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.
- 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 client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment.
- The data transfer from the client to the server stops. The server 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, the data can travel from the server to the client and acknowledgments can travel from the client to the server but the client cannot send any more data segments to the server.
- The second segment (ACK) consumes no sequence number. Although 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.

## CONGESTION

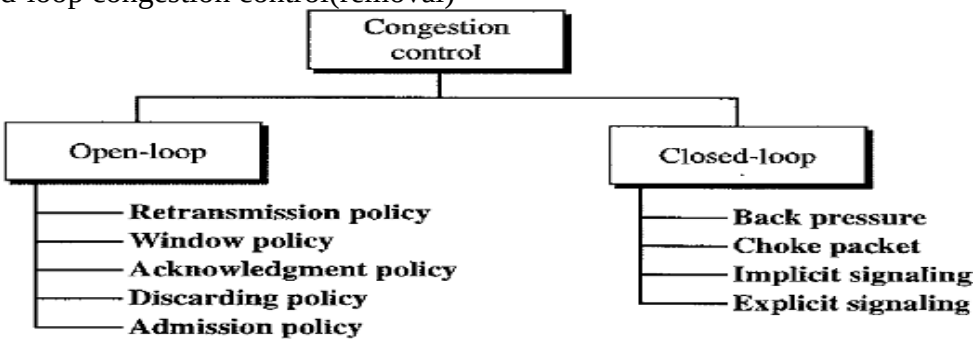
Congestion in a network may occur if the **load** on the network is greater than the *capacity* of the network.

## CONGESTION CONTROL

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

Congestion control mechanisms can be divided into two categories:

1. Open-loop congestion control(prevention)
2. Closed-loop congestion control(removal)




### Open-Loop Congestion Control

In open-loop congestion control, policies are applied to prevent congestion before it happens. Here congestion control is handled by either the source or the destination.

#### Retransmission Policy

- The packet needs to be retransmitted by sender, when a packet is lost or corrupted.
- Retransmission is sometimes unavoidable. It 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.

Example: Retransmission policy used by TCP is designed to prevent or alleviate congestion.

#### Window Policy

- 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 is expired several packets will be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse.
- The Selective Repeat window tries to send the specific packets that have been lost or corrupted.

#### Acknowledgment Policy

- The acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.
- 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 acknowledgment only if it has a packet to be sent or a special timer expires.

#### Discarding Policy

- A good discarding policy by routers may prevent congestion.

- Example: In audio transmission if the policy is to discard less sensitive packets when congestion happens, the quality of sound is still preserved and congestion is prevented.

**Admission Policy**

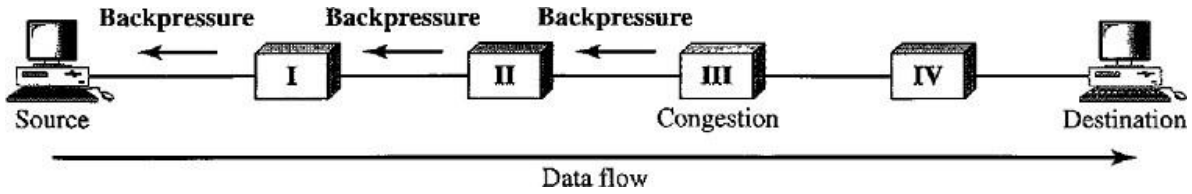
- An admission policy can prevent congestion in virtual-circuit networks.
- Switches first check the resource requirement of a data flow before admitting it to the network.
- A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

**Closed-Loop Congestion Control**

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols are: Back pressure, Choke packet, Implicit signaling, Explicit signaling.

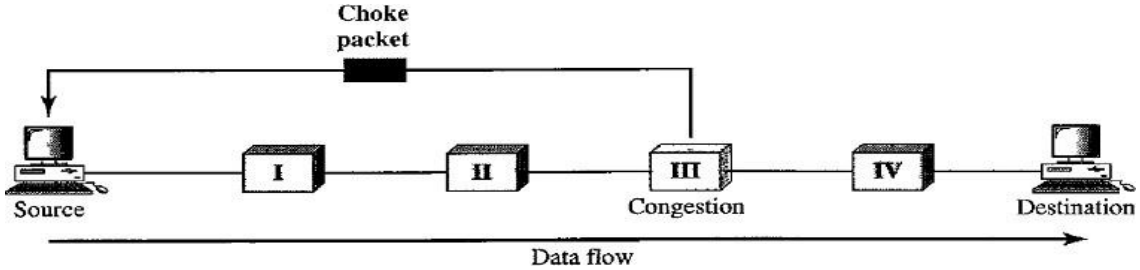
**Backpressure**

- In Backpressure mechanism, a congested node stops receiving data from the immediate upstream node.
- This may cause the upstream nodes to become congested and they reject data from their upstream nodes.
- Backpressure is a node-to-node congestion control that starts with a node and propagates in the opposite direction of data flow to the source.
- The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a data flow is coming.




**Choke Packet**

- A choke packet is a packet sent by a node to the source to inform that congestion has occurred.
- In the choke packet method, the warning is sent from the router, which has encountered congestion to the source station directly. The intermediate nodes through which the packet has traveled are not warned.




**Implicit Signaling**

- In implicit signaling, there is no communication between the congested nodes and the source.
- Source guesses that there is congestion somewhere in the network from other symptoms.

Example: when a source sends several packets and there is no acknowledgment for a while,

one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network and the source should slow down sending speed.

**Explicit Signaling**

- The node that experiences congestion can explicitly send a signal to the source or destination.
- In explicit signaling method, the signal is included in the packets that carry data.
- Explicit signaling can occur in either the forward or the backward direction.
- **Backward Signaling** A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.
- **Forward Signaling** A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments to get rid of the congestion.

## QUALITY OF SERVICE (QoS)

It is an internetworking issue. There are four Flow characteristics are related to Qos.
They are: Reliability, Delay, Jitter and Bandwidth.

**Reliability:** Lack of reliability means that losing a packet or acknowledgment, which entails retransmission. Electronic mail, file transfer, and Internet access have reliable transmissions.

**Delay:** Applications can tolerate delay in different degrees. Telephony, audio conferencing, video conferencing, and remote log-in need minimum delay.

**Jitter:**It is the variation in delay for packets belonging to the same flow. If four packets depart at times 0, 1, 2,3 and arrive at 20, 21, 22, 23 all have the same delay = 20 units of time. Audio and video applications accept the delay of packets as long as if the delay is same for all the packets.

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

## TECHNIQUES TO IMPROVE QoS

There are four techniques that will improve the QoS:

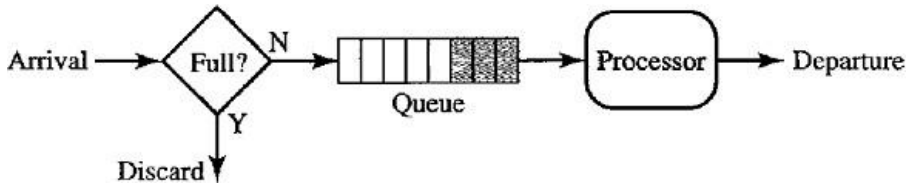
1. Scheduling
2. Traffic shaping
3. Admission control
4. Resource reservation

### Scheduling

Packets from different flows arrive at a switch or router for processing.
Several scheduling techniques are designed to improve the quality of service such as: FIFO Queuing, Priority Queuing and Weighted fair queuing.

### First-In-First-Out Queuing (FIFO)

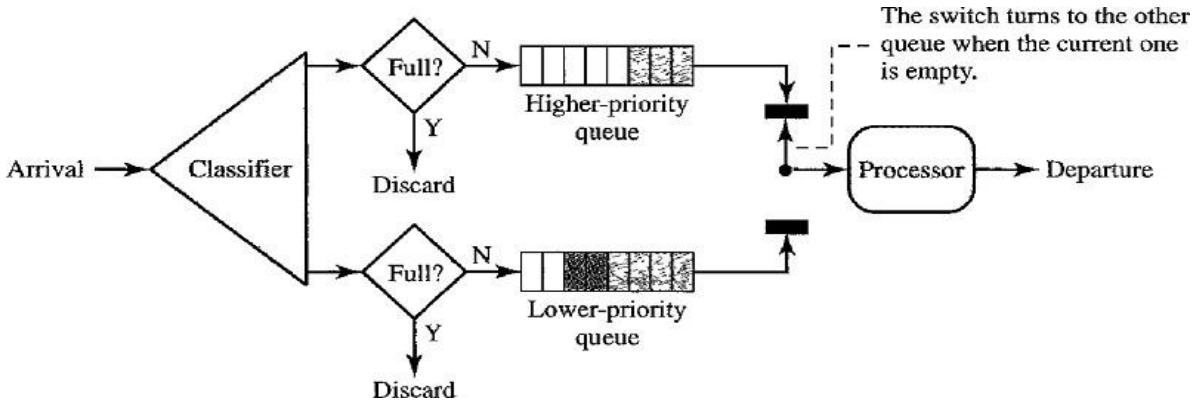
- In FIFO queuing, the packets wait in a buffer (queue) until the node (router or switch) is ready to process them.
- If the average arrival rate is higher than the average processing rate then the queue will fill up and new packets will be discarded.
- That means the speed of receiving the packets in to buffer is more than the speed of processing the packets by processor then the buffer will be filled completely and then there is no place for newly arrived packets hence these packets will be discarded.




### Priority Queuing

- In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue.
- The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last.
- The system does not stop serving a queue until it is empty.

- There is drawback with priority queues called **Starvation** (i.e.) if there is a continuous data flow in a high-priority queue, the packets in the low-priority queues will never have a chance to be processed.

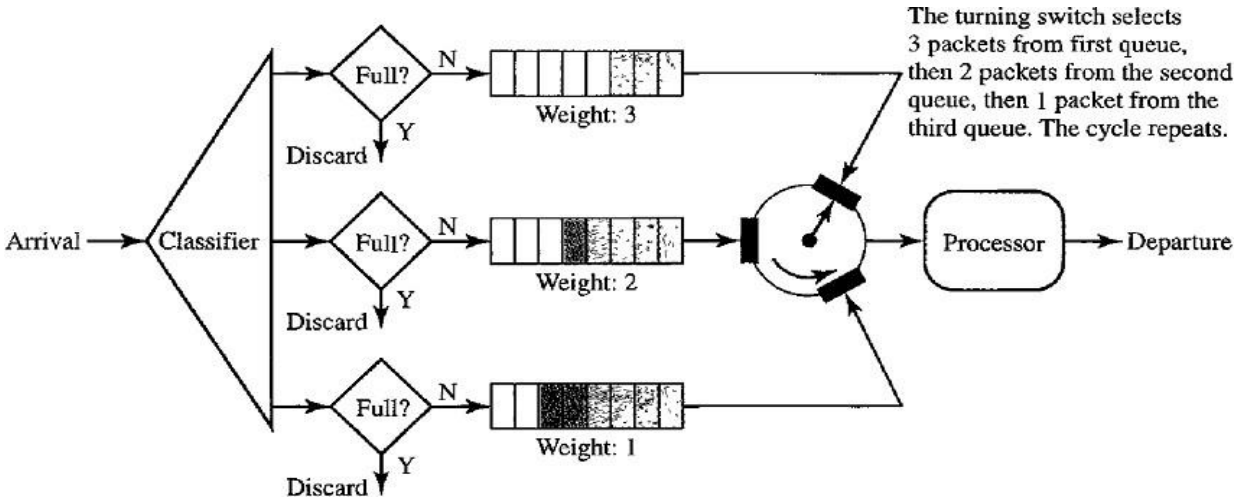



**Weighted Fair Queuing**

- In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues 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.

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 will achieve fair queuing with priority.

**Traffic Shaping**

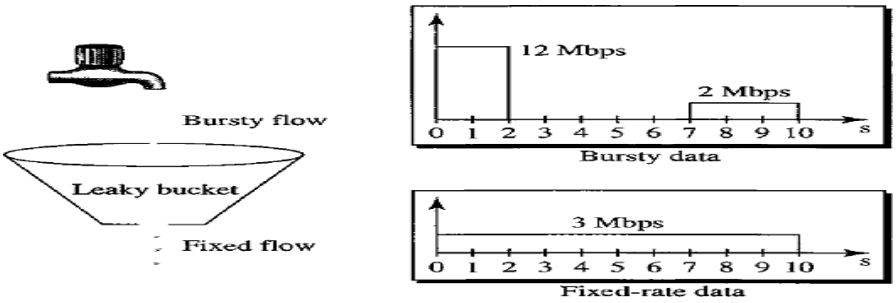



Traffic shaping is a mechanism to control the amount of traffic and the rate of the traffic sent to the network. Two techniques can shape traffic: **Leaky bucket** and **Token bucket.**

**Leaky Bucket**

- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant.

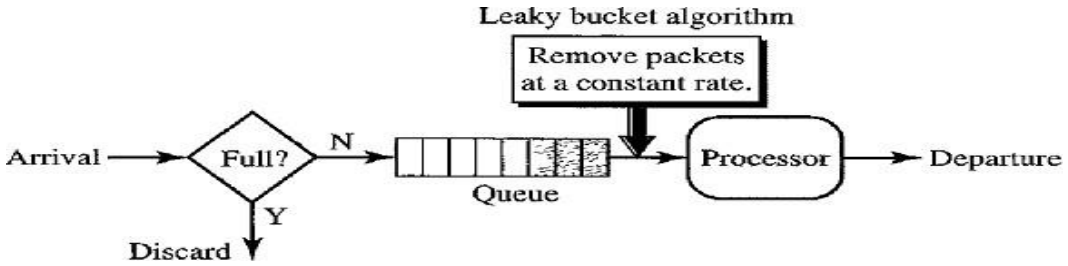
- In networking, a technique called leaky bucket can smooth out Bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.




- In the above figure, 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.
- The host sends a burst of data at a rate of 12 Mbps for 2 sec, for a total of 24 M bits of data.
- The host is silent for 5 sec and then sends data at a rate of 2 Mbps for 3 sec, for a total of 6 M bits of data.
- In total the host has sent 30 M bits of data inl0s.
- The leaky bucket smoothen the traffic by sending out data at a rate of 3 Mbps during the same 10sec.
- Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host.

This way the leaky bucket may prevent congestion.

Consider the below figure that shows implementation of Leaky Bucket:




- A FIFO queue holds the packets. If the traffic consists of fixed-size packets 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 orbits.

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.

**Problems with Leaky Bucket**

1. The leaky bucket is very restrictive. If a host is not sending for a while, its bucket becomes empty.
2. After some time 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.

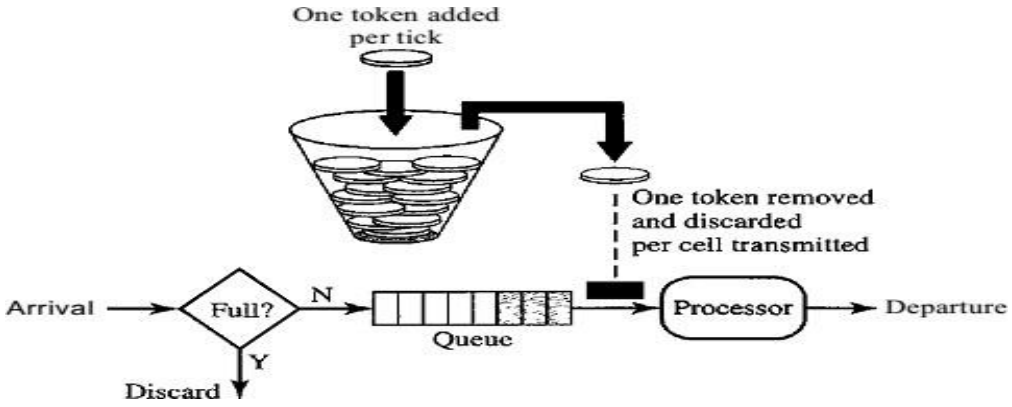
These problems can be overcome by Token bucket algorithm.

**Token Bucket**

- 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.

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.
- 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 by1.
- Each time a unit of data is sent, the counter is decremented by1.
- When the counter is zero, the host cannot send data.

**Resource Reservation**

- A flow of data needs resources such as a buffer, bandwidth, CPU time, and soon.
- The quality of service is improved if these resources are reserved before data transfer.

**Admission Control**

- Admission control refers to the mechanism used by a router or a switch to accept or reject a flow based on predefined parameters called flow specifications.
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity and its previous commitments to other flows can handle the new flow.

Note: Capacity is in terms of bandwidth, buffer size, CPU speed, etc.

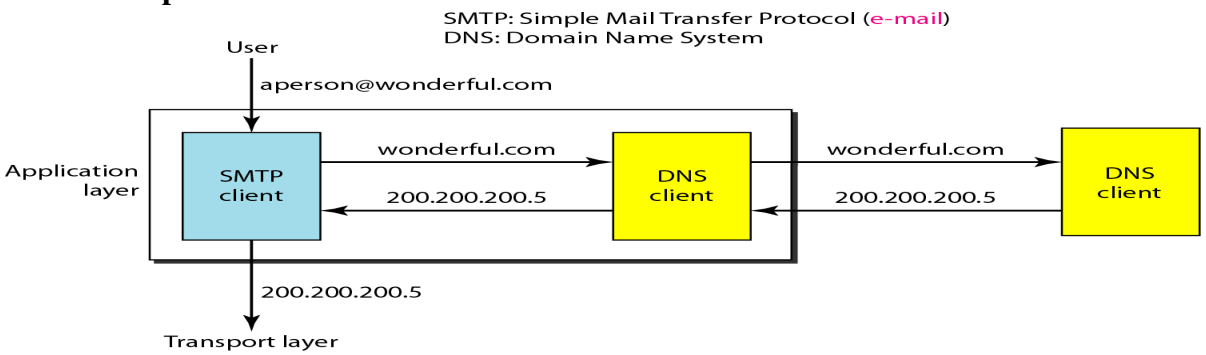
## DOMAIN NAME SYSTEM

The client/server programs can be divided into two categories:

1. Programs that can be directly used by user.
2. Programs that support other application programs.

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

### Basic concept of DNS




The above figure shows a DNS client/server program can support an E-mail program to find the IP address of an E-mail recipient.

- A user of an E-mail program knows the E-mail address of the recipient but the IP protocol needs the IP address.
- The DNS client program sends a request to a DNS server to map the E-mail address to the corresponding IP address.
- To identify an entity TCP/IP protocols uses the IP address, which uniquely identifies the connection of a host to the Internet.
- People prefer to use names instead of numeric addresses. Hence we need a system that can map a name to an address or an address to a name. DNS is designed for this purpose.

### NAMESPACE

The names assigned to machines must be unique because the addresses are unique.
A name space that maps each address to a unique name can be organized in two ways:

- Flat Name Space
- Hierarchical Name Space

### Flat Name Space

- In a flat name space, a name is assigned to an address.
- A name in this space is a sequence of characters without structure.

### Hierarchical Name Space

- In 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 etc.

**Example:**

Assume three Education institutions named one of their computers **Challenger**. The three colleges have given names by the central authority such as iitm.ac.in, berkeley.edu and smart.edu.

When these organizations add the name **Challenger** the names will be :

- challenger.iitm.ac.in
- challenger.berkeley.edu
- challenger.smart.edu

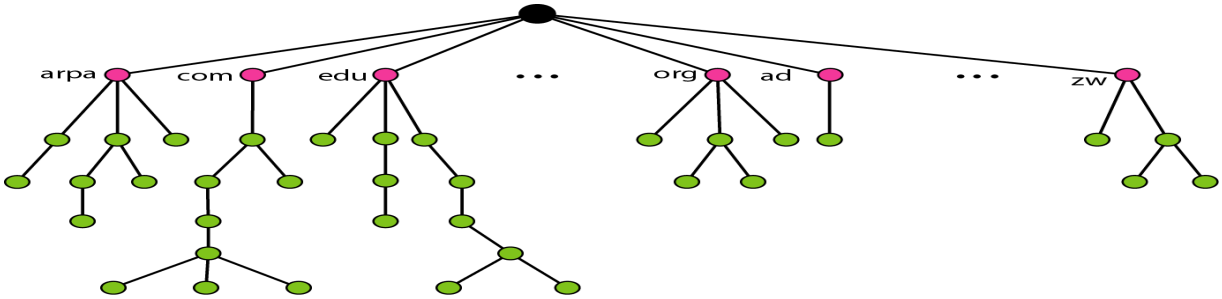
**DOMAIN NAME SPACE**

A domain name space was designed to have a Hierarchical Name Space.

In this design the names are defined in a tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.

**Label**

- Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string).
- DNS requires that children of a node have different labels, which guarantees the uniqueness of the domain names.

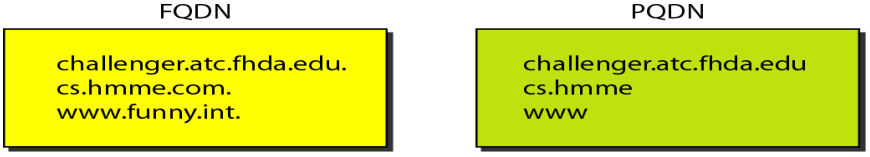



**Domain Name**

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

**Fully Qualified Domain Name (FQDN)**

- If a label is terminated by a null string, it is called a fully qualified domain name(FQDN).
- Example:**challenger.atc.fhda.edu.**

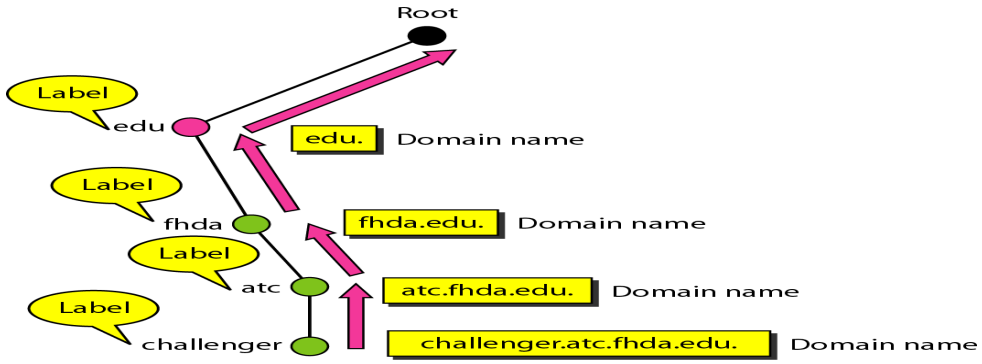



**Partially Qualified Domain Name (PQDN)**

- If a label is not terminated by a null string, it is called a PQDN.

- Example:**challenger**

If a user at the **"fhda.edu."** site wants to get the IP address of the challenger computer, he or she can define the partial name "**challenger**".
The DNS client adds the suffix **"atc.fhda.edu."** before passing the address to the DNS server.



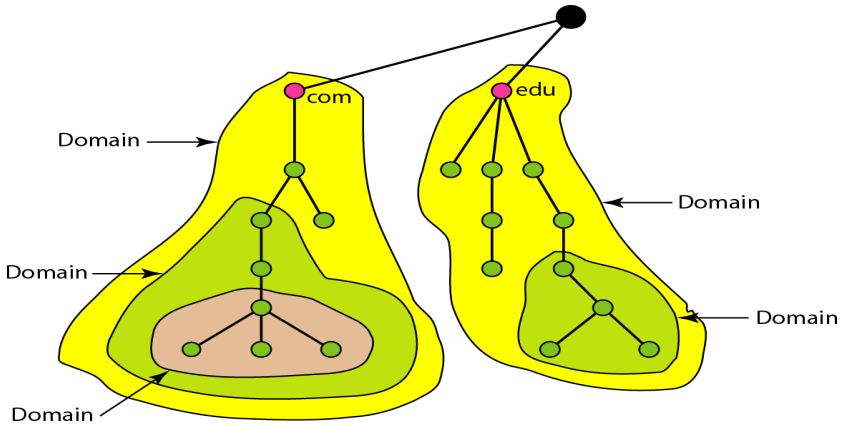

**Note:** The DNS client normally holds a list of suffixes such as:

- **atc.fhda.edu**
  - **fhda.edu**
  - **null**

These suffixes were added when the user defines an FQDN.

**Domain**

A **domain** is a sub-tree of the domain name space. The name of the domain is the domain name of the node at the top of the sub tree. A domain may itself be divided into sub-domains.




## DNS IN THE INTERNET

In the Internet the domain name space tree is divided into three different sections:

- Generic Domains,
- Country Domains
- The Inverse Domain

**Generic Domains**

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

name space database.

Generic domain labels are listed as:

| Label | Description |
|---|---|
| com | Commercial organizations |
| org | Nonprofit organizations |
| net | Network support centers |
| edu | Educational institutions |
| gov | Government institutions |

**Country Domains**

- The country domains section uses two-character country abbreviations. Ex: in for India, us for USA.
- Second labels can be organizational, or they can be more specific, national designations. Ex: .ac.in for nptel.ac.in, .gov.in for tspsc.gov.in etc.

**Inverse Domain**

The inverse domain is used to map an Address to a Name. It uses inverse query or pointer query.

**DISTRIBUTION OF NAME SPACE:**

**The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. In this section, we discuss the distribution of the domain name space.**

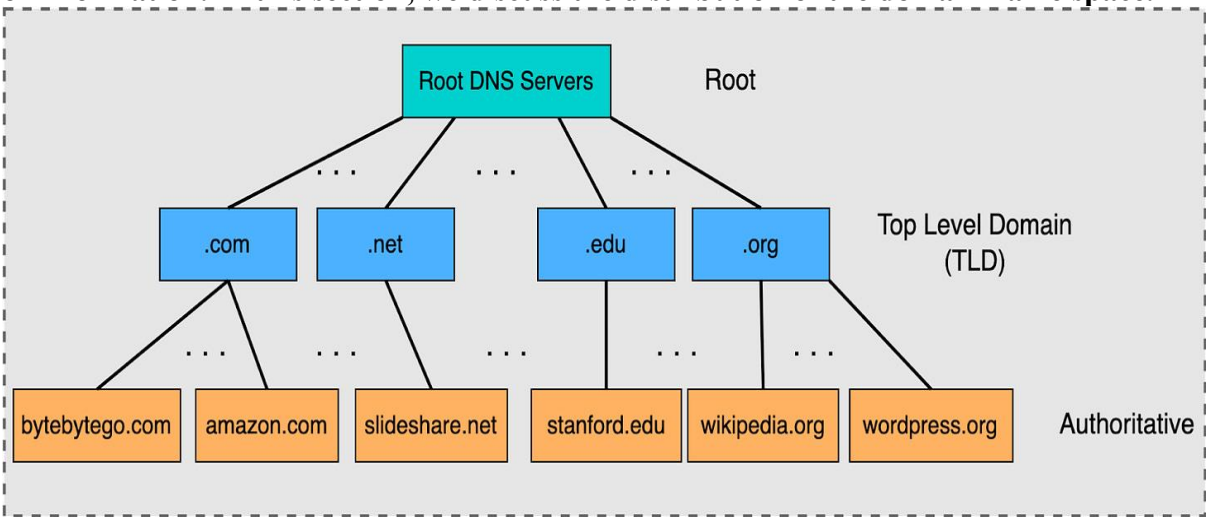
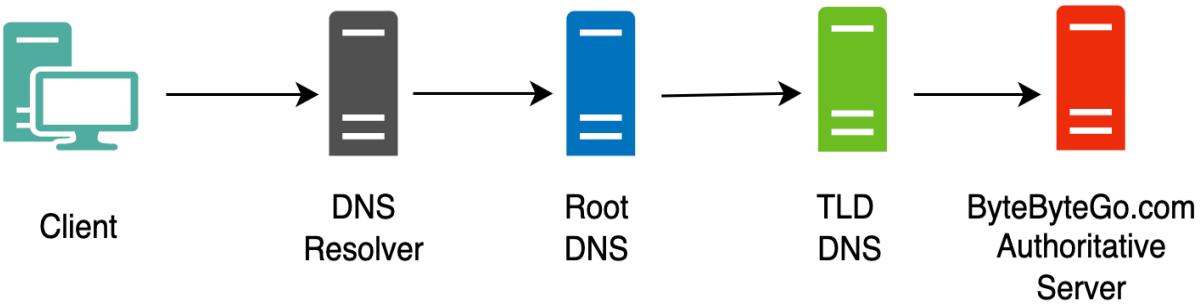



**Figure :** ***Hierarchy of name servers***



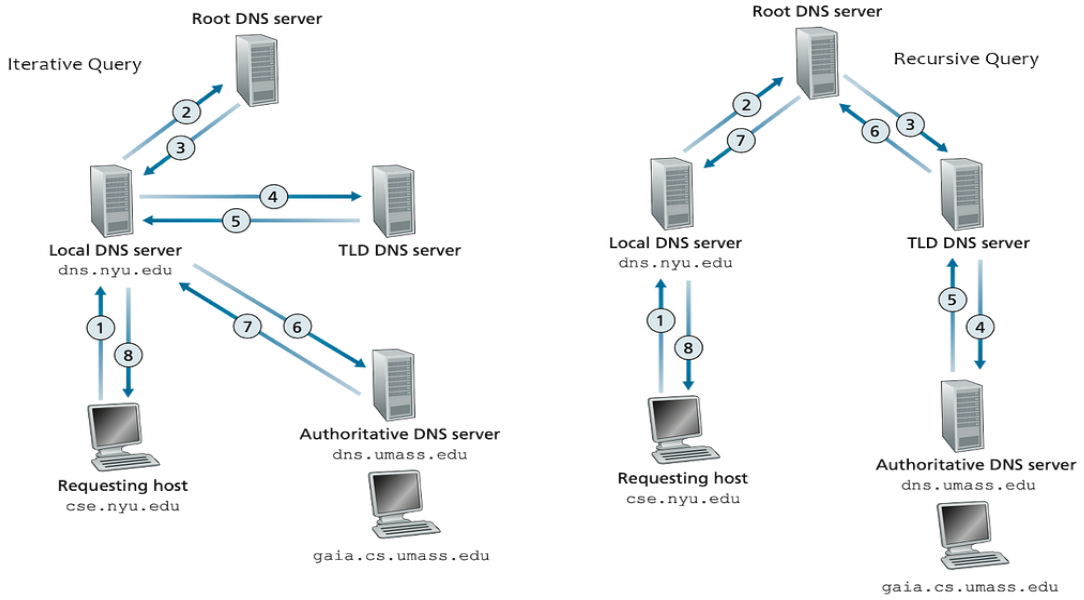

## RESOLUTION

Mapping a name to an address or an address to a name is called Name-Address Resolution.

**Types of Resolution:**

A recursive DNS lookup is where one DNS server communicates with several other DNS servers to hunt down an IP address and return it to the client.

This is in contrast to an iterative DNS query, where the client communicates directly with each DNS server involved in the lookup.



## REMOTE LOGGING

In the Internet, users may want to run application programs at a remote site and create results that can be transferred to their local site.

**Example:** Students may want to connect to their university computer lab from their home to access application programs for doing homework assignments or projects.

A General purpose client/server program that allows a user to log-on to a remote computer to access any application program on that remote computer.

After logging on, a user can use the available services on the remote computer and transfer the results back to the local computer.

### TELNET (TErminaL NETwork)

TELNET is a client/server application program. It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO).

TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

### Timesharing Environment

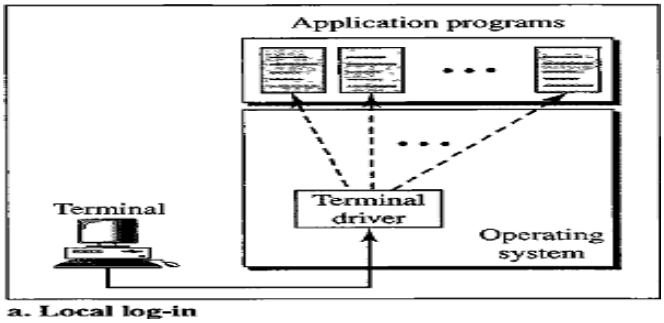
TELNET works in Time Sharing Environment. The interaction between a user and the computer occurs through a terminal.

### Logging

- In a timesharing environment, users are part of the system with some right to access resources. Each authorized user has Identification (User ID) and a password.
- 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.

### Local log-in

- When a user logs into a local timesharing system it is called local log-in.
- As a user types at a terminal the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system.
- The operating system interprets the combination of characters and

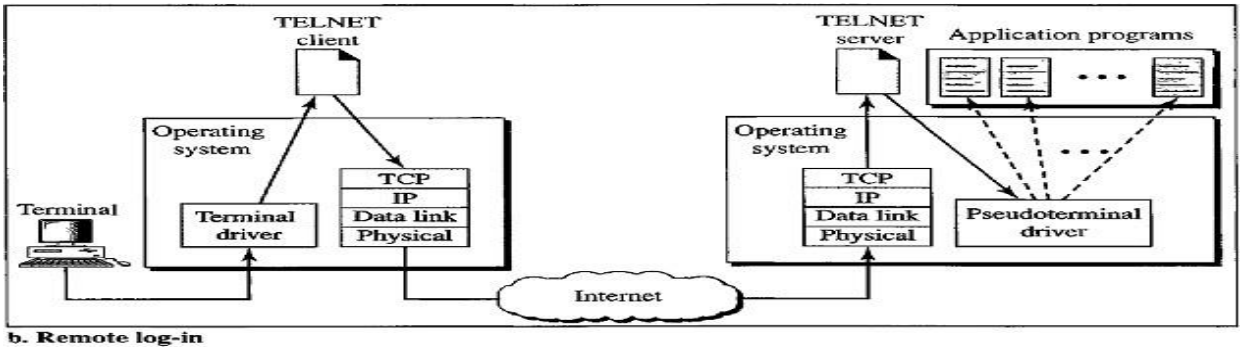



a. Local log-in

invokes the desired application program or utility.

**Remote Log-in**

- When a user wants to access an application program or utility located on a remote machine, the user performs remote log-in. TELNET uses client and server programs.
- The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters to a universal character set called Network Virtual Terminal (NVT) characters and delivers them to the local TCP/IP protocol stack.
- The commands or text in NVT form travel through the Internet and arrive at the TCP/IP stack at the remote machine.



b. Remote log-in

- The characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.
- The characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server. It is designed to receive characters from a terminal driver.
- Hence the characters can be passed to **Pseudo-terminal driver** which passes the characters to operating system.
- The operating system then passes the characters to the appropriate application program.

**Network Virtual Terminal (NVT)**

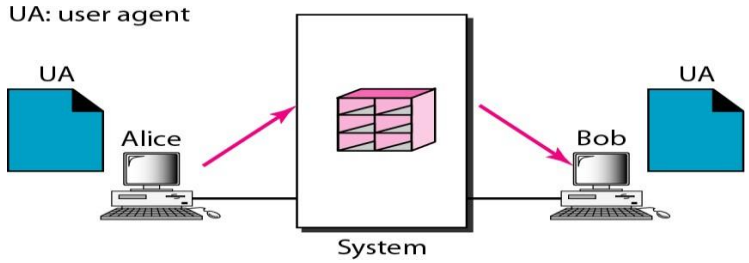
- TELNET defines a universal interface called the Network Virtual Terminal character set. Via this interface the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network.
- The server TELNET translates data and commands from NVT form into the form acceptable by the remote computer.

## ELECTRONIC MAIL

Electronic mail (E-mail) is one of the most popular Internet services. E-mail allows a message to include text, audio, and video. There are Four Scenarios of E-mail:

**First Scenario**

- In the first scenario, the sender and the receiver of the E-mail are user application programs on the same system. They are directly connected to a shared system.
- The administrator has created one mailbox for each user where the received messages are stored.
- A mailbox is part of a local hard drive and it a special file with permission restrictions. Only the owner of the mailbox has access to it.

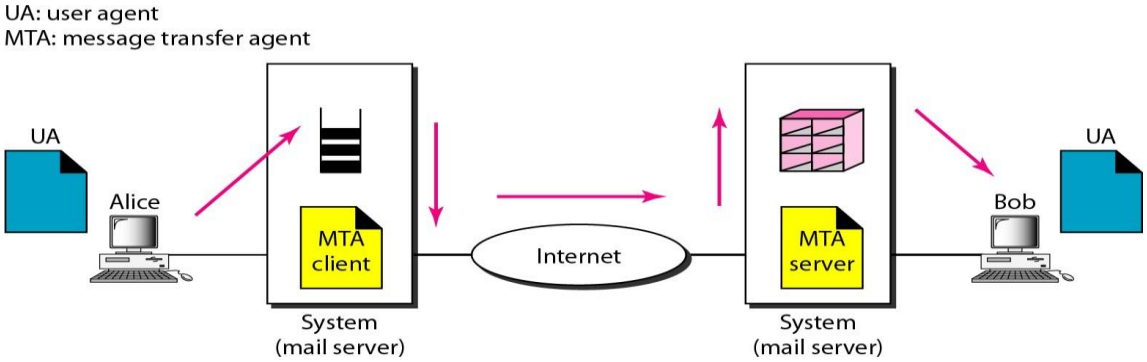



Example: Consider the above figure, Alice and Bob are two users of mail server.

- When a user Alice needs to send a message to Bob, Alice runs a User Agent (UA) program to prepare the message and store it in Bob's mailbox.
- The message has the sender and recipient mailbox addresses (names offiles).
- Bob can retrieve and read the contents of his mailbox using a User Agent.

**Second Scenario**

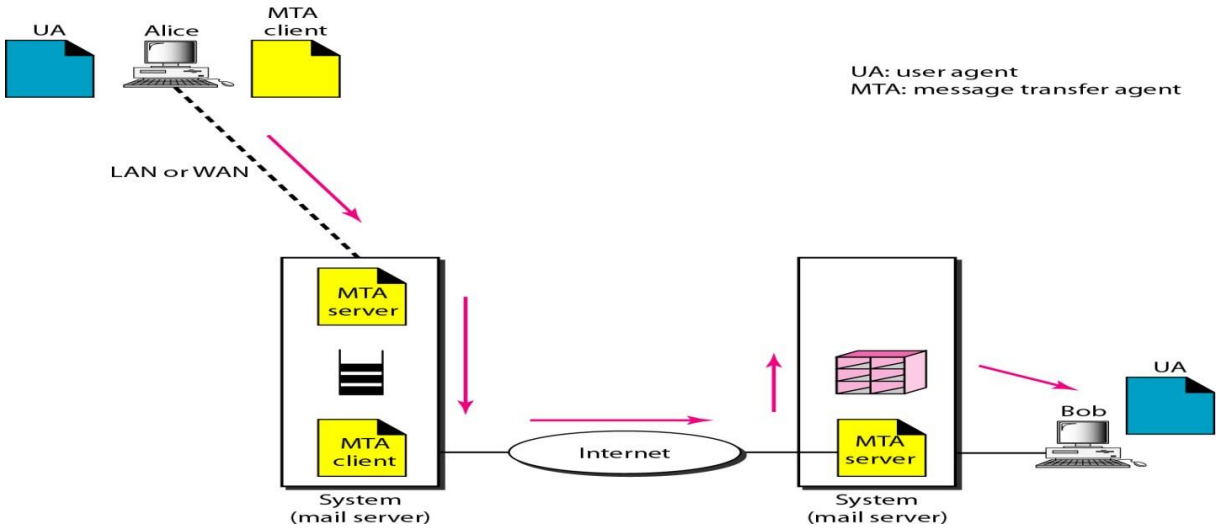
- In the second scenario, the sender and the receiver of the E-mail are user application programs on two different systems. The message needs to be sent over the Internet.
- We need User Agents (UAs) and Message Transfer Agents(MTA's).




- Alice needs to use a user agent program to send her message to the mail server at her own site.
- The mail server at Alice site uses a queue to store messages waiting to be sent.
- Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site.

- The message needs to be sent through the Internet from Alice's site to Bob's site. Here two Message Transfer Agents are needed: one for client and one for server.
- Most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection.
- The client can be alerted by the system when there is a message in the queue to be sent.

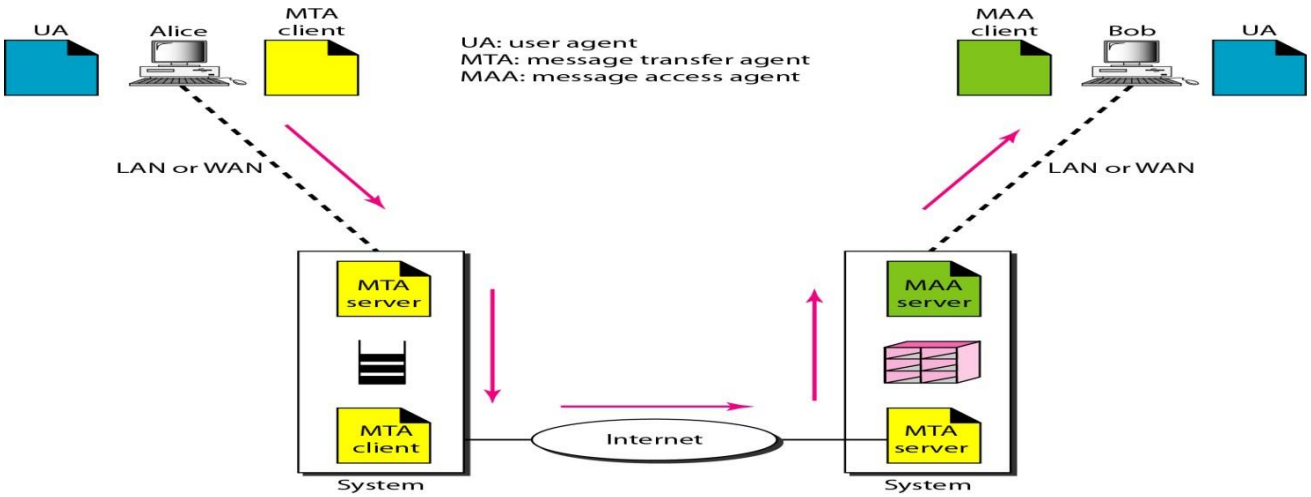
**Third Scenario**




- In the third scenario, Bob is directly connected to his system (i.e. Mail Server). Alice is separated from her system.
- Alice is connected to the mail server via WAN or LAN.
- In an organization that uses one mail server for handling E-mails, all users need to send their messages to this mail server.
- User agent of Alice prepares message and sends the message through the LAN or WAN.
- Whenever Alice has a message to send, Alice calls the user agent and user agent calls the MTA client.
- The MTA client establishes a connection with the MTA server on the system.
- The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site. The system receives the message and stores it in Bob's mailbox.
- Bob uses his user agent to retrieve the message and reads it.
- It needs two MTA client and two MTA server programs.

**Fourth Scenario**

- It is the most common scenario, Alice and Bob both are connected to their mail server by a WAN or a LAN.
- After the message has arrived at Bob's mail server, Bob needs to retrieve it. Now Bob needs another set of client/server agents called Message Access Agents (MAA). Bob uses an MAA client to retrieve his messages.
- The client sends a request to the MAA server and requests the transfer of the messages.



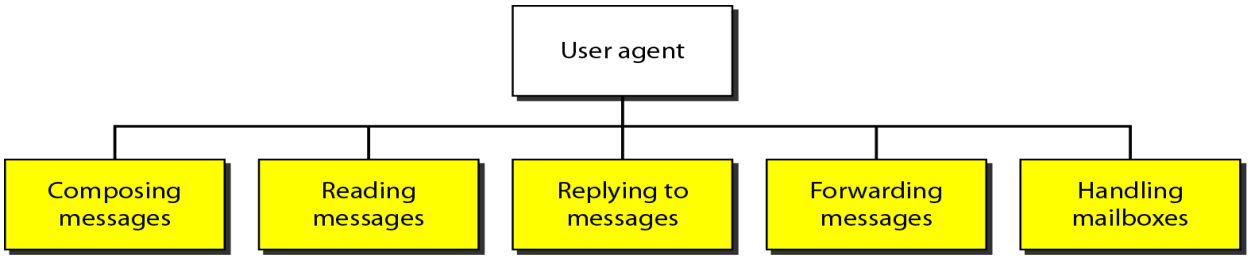

**Architecture of E-mail**

There are three major components in the architecture of E-mail:

1. User Agent
2. Message Transfer Agent
3. Message Access Agent

**User Agent**

User Agent provides services to the user to make the process of sending and receiving a message easier. Services provided by User agent are:




**Composing Messages**

A user agent helps the user to 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.

**Reading Messages**

The user agent reads the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Each E-mail contains the following fields:

1. A number field.
2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to.
3. The size of the message.
4. The sender.
5. The optional subject field.

**Replying to Messages**

After reading a message, a user can use the user agent to reply to a message. A user agent allows the user to reply to the original sender or to reply to all recipients of the message.

**Forwarding Messages**

It means sending a message to a third party. A user agent allows receiver to forward message.

**Handling Mailboxes**

- A user agent normally creates two mailboxes: **Inbox** and **Outbox**.
- Inbox and Outbox is a file with a special format that can be handled by user agent.
- **Inbox** keeps all the received E-mails until they are deleted by the user.
- **Outbox** keeps all the sent E-mails until the user deletes them.

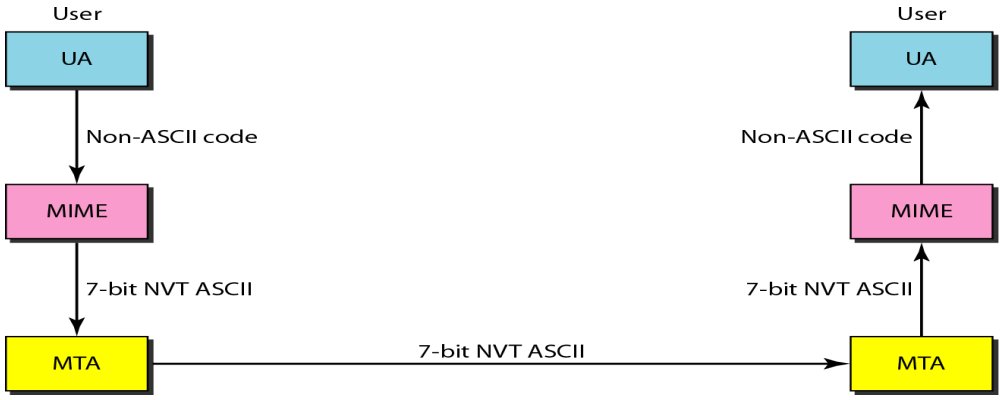
**Mailing List**

- Electronic mail allows one name (an alias) to represent several different E-mail addresses is called a mailing list.
- Every time a message is to be sent, the system checks the recipient's name against the alias database.

**MIME (Multipurpose Internet Mail Extensions)**

- MIME is a supplementary protocol that allows non-ASCII data to be sent through E-mail.
- French, German, Hebrew, Russian, Chinese, and Japanese are non-ASCII characters.
- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet.

The message at the receivi



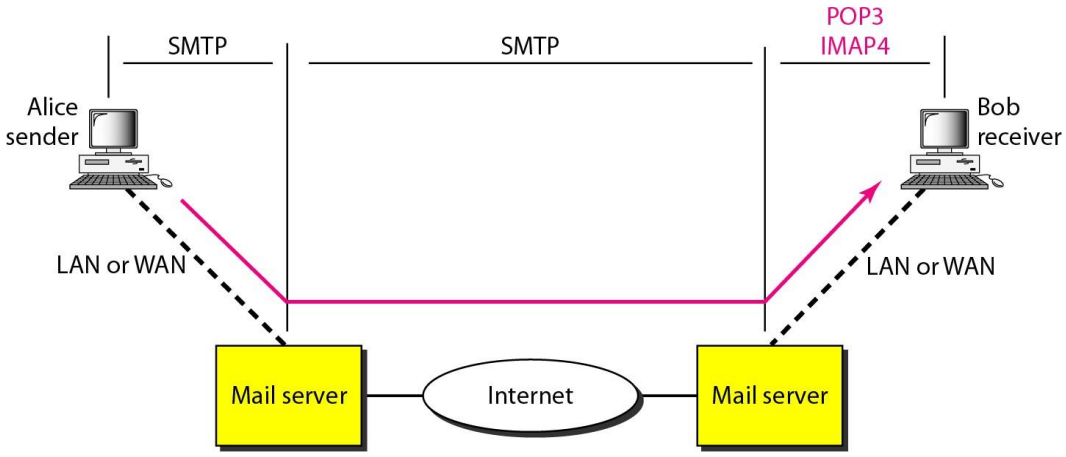
## SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

**Message Transfer Agent: SMTP**

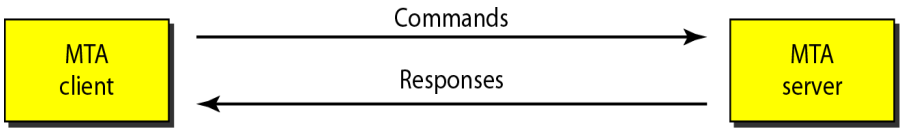
- The actual mail transfer is done through message transfer agents (MTA).
- Simple Mail Transfer Protocol defines the MTA Client and MTA server in the internet.
- MTA Client is used to send mail and MTA Server is used

to receive a mail. SMTP is used two times:

1. Between sender and sender mail server.
2. Between sender mail server and receiver mail server.




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 end-of-line token.




**Commands**

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

There are five mandatory commands are used. Every implementation must support these five commands: HELO (Sender's Host name), MAIL FROM, RCPT TO, DATA, QUIT.

**Responses**

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

Responses are divided into four categories: The leftmost digit of the code 2, 3, 4, and 5 defines the category.

- 2-Positive Completion Reply
- 3-Positive Intermediate Reply
- 4-Transient Negative Completion Reply
- 5-Permanent Negative Completion Reply

**Mail Transfer Phases**

The process of transferring a mail message occurs in three phases:

- Connection Establishment
- Mail Transfer
- Connection Termination

## FILE TRANSFER PROTOCOL (FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another. FTP uses the services of TCP. FTP implemented to solve below problems.

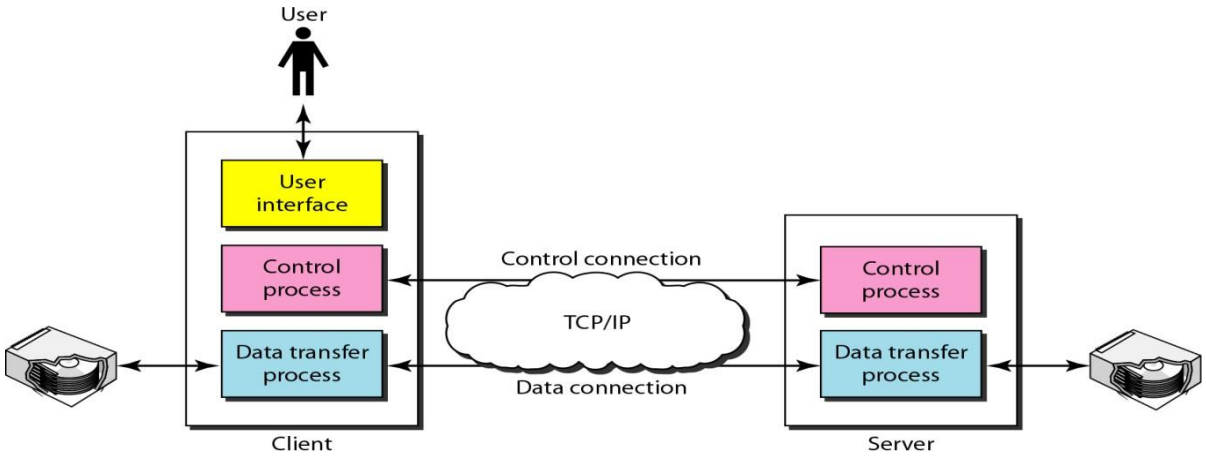
**Problems with File transfer**

- 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.

FTP differs from other client/server applications in that it establishes two connections between the hosts. FTP uses two well-known ports these connections.

1. Data transfer connection (Port 20 is used)
2. Control connection (Port 21 is used)

**Basic Model of FTP**




Consider the above figure that shows basic model of FTP that contains client and server components.

- Client has three components: User Interface, Client Control Process, and Client Data Transfer Process.
- Server has two components: Server Control Process and Server Data Transfer Process.
- The control connection is made between the control processes.
- The data connection is made between the data transfer processes.
- The control connection remains connected during the entire interactive FTP session.
- The data connection is opened and then closed for each file transferred.
- When a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

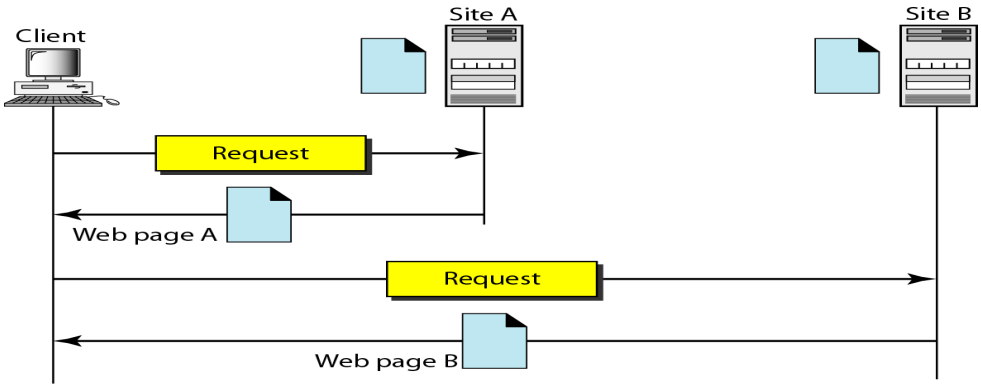
## WORLD WIDE WEB (WWW)

**World Wide Web** (WWW) is a repository of information linked together from locations all over the world. WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet.

### Architecture

The WWW is a distributed client/server service, in which a client using a browser can access a service using a server.

- The service provided is distributed over many locations called site.
- 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 is called Hyperlink.
- The pages can be retrieved and viewed by using browsers.




Architecture of WWW contains four parts: **1. Client 2. Server**
**3.URL 4. Cookies Client(Browser)**

A Client is a browser that interprets and displays a Web document.
Each browser consists of three parts: **Controller**, **Client protocol**, and **Interpreters.**

- The controller receives input from the keyboard or the mouse and uses the client programs to access the document.
- After the document has been accessed, the controller uses one of the interpreters to display the document on the screen.
- The interpreter can be HTML, Java, or JavaScript depending on the type of document.
- The client protocol can be FTP or HTTP.

**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.
- A server uses multi-threading or multi-processing for answering more than one request at a time to increase the efficiency.

**Uniform Resource Locator (URL)**

The Uniform Resource Locator (URL) is a standard for specifying any kind of information on the Internet. A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators.

URL defines four things: Protocol, Host computer, Port, and Path.


- **Protocol:** It is the client/server program used to retrieve the document. Ex:FTP or HTTP.
- **Host:** The host is the computer on which the information is located. Web pages are usually stored in computers and computers are given **alias names** that usually begin with the characters "www". This is not mandatory.
- **Port:** The URL can optionally contain the port number of the server.
- **Path:** It is the pathname of the file where the information is located.

Note: The path can itself contain slashes that separate the directories from the subdirectories and files.

**Cookies**

Cookies are used to devise the following functionalities:

- Some websites need to allow access to registered clients only.
- Websites are being used as electronic stores (such as Flipkart or Amazon) that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
- Some websites are used as portals: the user selects the Web pages he wants to see.
- Some websites are just advertising.

**Creation and Storage of Cookies**

The creation and storage of cookies depend on the implementation:

1. When a server receives a request from a client, it stores information about the client in a file or a string.

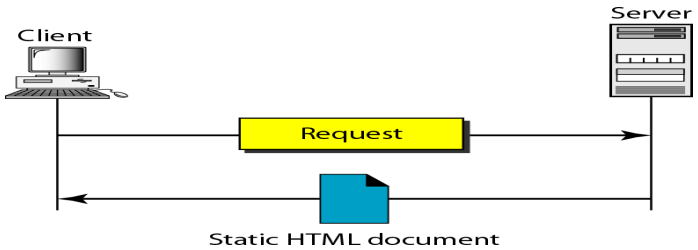
   The information may include the domain name of the client, a timestamp, the contents of the cookie such as client name, client registration number and other information depending on the implementation.
2. The server includes the cookie in the response that it sends to the client.
3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the domain server name.

**WEB DOCUMENTS**

Documents in the WWW can be grouped into three 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 (i.e.) the contents of the file are determined when the file is created, not when it is used.
- The contents in the server can be changed but the user cannot change them.

- When a client accesses the document, a copy of the document is sent and the user can then use a browsing program to display the document.




**Hypertext Markup Language (HTML)**

- Hypertext Markup Language (HTML) is a language for creating Web pages.
- Data for a Web page are formatted for interpretation by a browser.
- HTML allows us to embed formatting instructions in the file itself. The instructions are included with the text.

A Web page is made up of two parts: **Head** and **Body.**

**Head:** The head is the first part of a Web page. The head contains the title of the page and other parameters that the browser will use.

**Body:** The actual contents of a page are in the body, which includes the text and the tags. The text is the actual information contained in a page. The tags define the appearance of the document.

**HTML Tags**

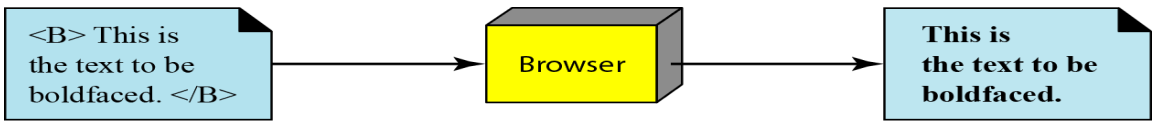
- Every HTML tag is a name followed by an optional list of attributes enclosed between less-than and greater-than symbols (< and>).
- An attribute is followed by an equals sign and the value of the attribute.
- The browser makes a decision about the structure of the text based on the tags, which are embedded into the text.


a. Beginning tag


b. Ending tag

The common tags used in HTML are : **Bold, Italic, Underline** the text.

- The two **Bold** face tags **<B>**and **</B>**are instructions for the browser.
- The two **Italic** tags **<I>**and **</I>**make the text italic
- The two **Underline** tags **<U>**and **</U>**put underline below the text.




There are two other tags used in HTML are: **Image** and **Hyperlink** tags.

**Image tag**

- Non-textual information such as digitized photos or graphic images is not a physical part of an HTML document.

- The image tag defines the address (URL) of the image to be retrieved. An image tag is used to point to the file of a photo or image.
- It also specifies how the image can be inserted after retrieval. Image tag contains attributes such as: SRC ,ALIGN.
- SRC (source) defines the source address
- ALIGN defines the alignment of the image

**<IMG SRC=”/bin/images/image1.gif” ALIGN=MIDDLE>**

**Hyperlink tag**

- Hyperlink tag is needed to link documents together. Any item such as word, phrase, paragraph or image can refer to another document through a mechanism called an **anchor**.
- The anchor is defined by <A ... > and </A> tags and the anchored item uses the URL to refer to another document.
- When the document is displayed, the anchored item is underlined, blinking, or boldfaced.
- The user can click on the anchored item to go to another document.
- The reference phrase is embedded between the beginning and ending tags.
- The beginning tag can have an attributes called HREF (Hyperlink Reference) defines the address (URL) of the linked document.

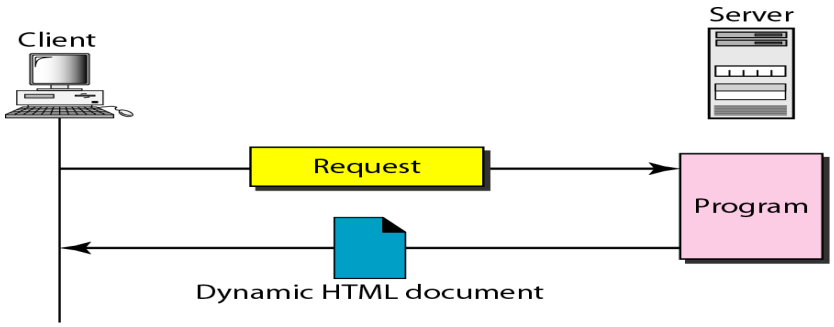
**<A HREF=**

**http://www/deanza.edu/forouzan> Author**

**</A> Dynamic Documents**

- A **dynamic document** is created by a Web server whenever a browser requests the document.
- When a request arrives, the Web server runs an application program that creates the dynamic document.
- The server returns the output of the program as a response to the browser.
- Because a fresh document is created for each request, the contents of a dynamic document can vary from one request to another.

Example: the retrieval of the time and date from a server is a dynamic document.




**Common Gateway Interface (CGI)**

- CGI is a technology that creates and handles dynamic documents.
- CGI is a set of standards that defines how a dynamic document is written, how data are input to the program and how the output result is used

- The CGI also defines a set of rules and terms that the programmer must follow.
- CGI allows programmers to use any of several languages such as C, C++, Bourne Shell, Korn Shell, C Shell, Tcl, or Perl.
- CGI program can be used to access resources such as databases, graphical packages etc.

**Input**

- The input from a browser to a server is sent by using a Form. If the information in a form is small (such as a word), it can be appended to the URL after a question mark.

Example: The following URL is carrying Form information (23, a value):

**http://www.deanza/cgi-bin/prog.pl?23**

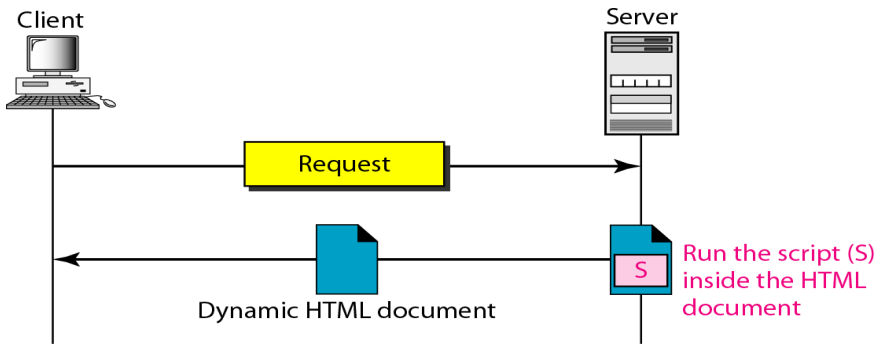
- If the input from a browser is too long to fit in the query string, the browser can ask the server to send a form. The browser can then fill the form with the input data and send it to the server.
- The information in the form can be used as the input to the CGI program.

**Output**

- CGI executes a CGI program at the server site and send the output to the client(browser).
- The output can be a plain text, graphics or binary data, a status code, instructions to the browser to cache the result, or instructions to the server to send an existing document instead of the actual output.
- The output of the CGI program always consists of two parts: **Header** and **Body**.
- The Header is separated by a blank line from the body.

**Scripting Technologies for Dynamic Documents**

- The problem with CGI technology is the inefficiency that results, if part of the dynamic document that is to be created is fixed and not changing from request to request.
- The solution is to create a file containing the fixed part of the document using HTML and embed a script, a source code that can be run by the server to provide the dynamic part.
- PHP, JSP, ASP, ColdFusion are the technologies have been involved in creating dynamic documents using scripts.




**Active Documents**

- Applications need a program or a script to be run at the client site. These are called active documents.

- When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client site(browser).

Example: Suppose we want to run a program that creates animated graphics on the screen. The program definitely needs to be run at the client site where the animation takes place.

**Java Applets**

- By using java applets we can create an active document.
- 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.
- Java can also be a stand-alone program that doesn't use a browser.
- An applet is a program written in Java on the server. It is compiled and ready to be run.
- The document is in byte-code (binary) format.
- The client process (browser) creates an instance of this applet and runs it.

**JavaScript**

- Java script 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.
- JavaScript is a very high level scripting language developed for this purpose.

## HYPERTEXT TRANSFER PROTOCOL (HTTP)

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP uses the services of TCP on well-known port 80.

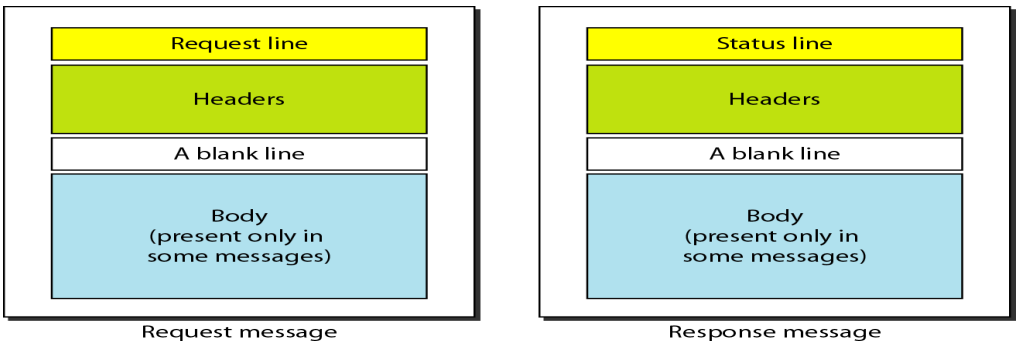
HTTP functions as a combination of FTP and SMTP.

**HTTP Transaction**

HTTP is a stateless protocol even though it uses TCP services. The client initializes the transaction by sending a request message. The server replies by sending a response.

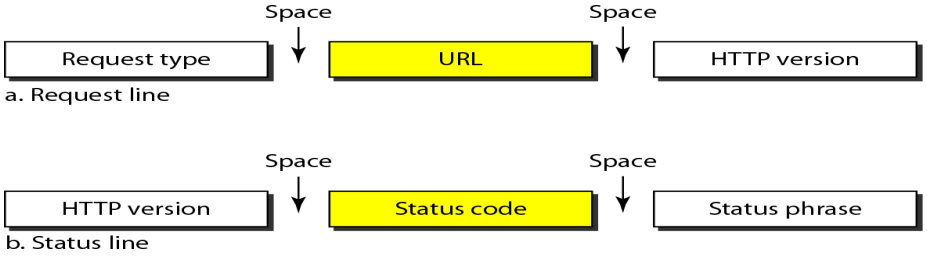
HTTP uses two types of messages: Request, Response

- A request message consists of a request line, a header, and optional body.
- A response message consists of a status line, a header, and optional body.




**Request and Status Lines**

- The first line in a request message is called a request line.
- The first line in the response message is called the status line.




**Request type**

This field is used in the request message. In version 1.1 of HTTP defines several request types. The request type is categorized into methods.

| Methods | Action |
|---|---|
| GET | Requests a document from the server |
| POST | Sends some information from the client to the server |
| PUT | Sends a document from the server to the client |

**URL:** By using URL, clients can access the webpage.

**Version** The most current version of HTTP is 1.1.

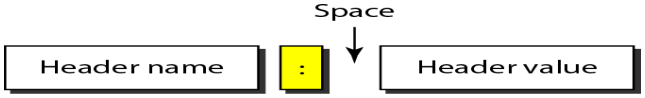
**Status code** is used in the response message. It consists of 3-digits. 100, 200, 300, 400, 500.

**Status phrase:** It explains the status code in text form, it is used in the response message.

| Status Code | Status phrase | Description |
|---|---|---|
| 100 | Continue, switch | Informational |
| 200 | OK, CREATED, ACCEPTED | Successful request |
| 300 | Moved Permanently or temporarily, Not modified. | Redirect Client to another URL |
| 400 | 400- Bad request, 404- Not found, | Error at client side |
| 500 | 500- Internal server error 503-Service unavailable | Error at server side |

**Header**

The header exchanges additional information between the client and the server. The header can consist of one or more header lines.



A Header line can be divided into 4 categories: 1. **General** 2.**Request** 3.**Response** 4.**Entity**.

1. A request message can contain Request header, General header, Entity header.
2. A response message can contain Response header, General header, Entity header.

**General header**

General header gives general information about the message such as Date, MIME version.

**Request header**

Request header specifies the client's configuration and the client's preferred documentformat. Example: Accept: Shows the medium format the client can accept

From: Shows the E-mail address of the user Host: Shows the host and port number of the server Referrer: Specifies the URL of the linked document User agent: Identifies the client program.

**Response header**

This header specifies the server's configuration and special information about the request. Example: Age: shows the age of the document, public: shows the supported list of methods, server: shows the server name and address.

**Entity header**

The entity header gives information about the body of the document. some request messages such as POST or PUT methods may contain a body also use this type of header.

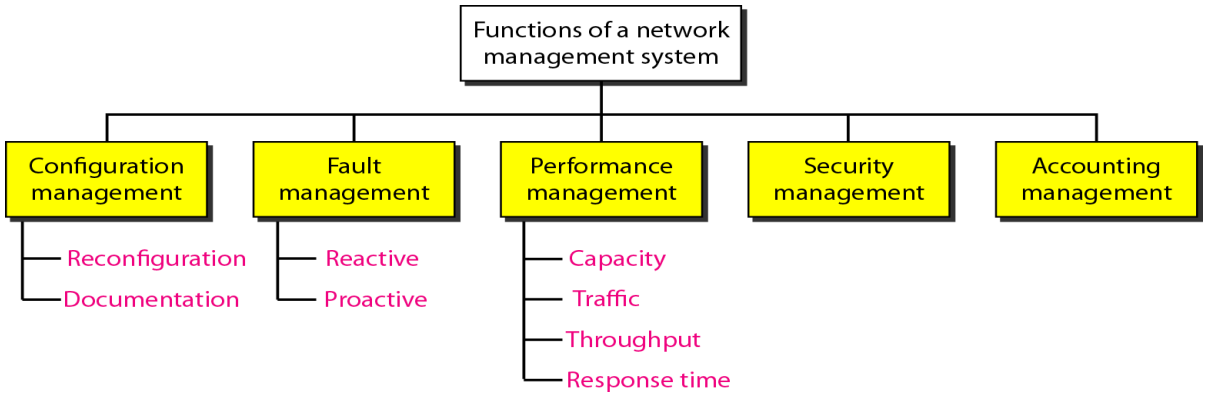
Examples: E tag- Gives an entity tag, Content-type- Specifies the medium type, Last- modified- Gives the date and time of the last change etc.

**Body**

The body can be present in a request or response message. Body contains the document to be sent or received.

## NETWORK MANAGEMENT: SNMP

Network Management can be defined as monitoring, testing, configuring and troubleshooting network components to meet a set of requirements defined




by an organization.

**Functions of Network Management System**

Network Management System can be divided into five broad categories:

**Configuration Management**

- A large network is usually made up of hundreds of entities that are physically or logically connected to one another.
- These entities have an initial configuration when the network is set up, but can change with time.
- The configuration management system must know the status of each entity and its relation to other entities.

Configuration management can be divided into two subsystems: Reconfiguration and Documentation.

**Reconfiguration**

Reconfiguration means in a large network, the network components and features are adjusted daily. There are three types of reconfiguration:

i. **Hardware reconfiguration:** It covers all changes to the hardware. These are handled manually. Example: A sub network (Router) may be added or removed from the network.

ii. **Software reconfiguration:** It covers all changes to the software. Most of the software reconfiguration can be automated. Example: Updating Operating system.

iii. **User-account reconfiguration:** It covers adding and deleting the users on a system and it also considers user's individual privileges and Group privileges.

Example: A user may have read and write permission with regard to some files, but only read permission with regard to other files.

**Documentation**

The network configuration and each subsequent change in hardware, software and user accounts must be documented.

**Hardware documentation**

- It involves two sets of documents: Maps andSpecifications.
- **Maps** track each piece of hardware and its connection to the network.
- General maps that shows the logical relationship as well as physical relationship between eachsub network.
- For each sub-network, there are one or more maps that show all pieces of equipment.
- **Specification** information such as hardware type, serial number, vendor address and phone number, time of purchase and warranty information must be included for each piece of hardware connected to the network.

**Software Documentation** It includes information such as the software type, the version, the time installed etc.

**User documentation** Operating system utilities allows the documentation of user accounts and their privileges. The information in these files are updated and secured.

**Fault Management**

Fault management is the area of network management that handles the issues in network components. Example: A fault may be a damaged communication medium.

Fault management system has two subsystems: Reactive and Proactive.

**Reactive Fault Management**

The responsibilities of reactive fault management can be divided into 4 steps:

i. **Detect the fault:** Fault management system must have to detect the exact location of the fault.
ii. **Isolate the fault:** If a fault is isolated that affects only a few users. After isolation, the affected users are immediately notified and given an estimated time of correction.
iii. **Correct the fault:** This may involve replacing or repairing the faulty components.
iv. **Record the fault:** After the fault is corrected, it must be recorded (i.e. documented). The record should show the exact location of the fault, the possible cause, the action or actions taken to correct the fault, the cost and time it took for each step.

**Proactive Fault Management**

Proactive fault management tries to prevent faults from occurring. Some failures can be predicted and prevented.

**Performance Management**

Performance management tries to monitor and control the network to ensure that it is running as efficiently. It can be measured by the following concepts: Capacity, Traffic, Throughput, Response time.

- **Capacity of the Network** Every network has a limited capacity, and the performance management system must ensure that it is not used above this capacity.

Example: If a LAN is designed for 100 stations at an average data rate of2 Mbps, it will not operate properly if 200 stations are connected to the network.

- **Traffic** Traffic can be measured in two ways: Internally and Externally.

Internal traffic is measured by the number of packets (or bytes) traveling inside the network.

External traffic is measured by the exchange of packets (or bytes) outside the network.

- **Throughput** It can be measured by an individual device such as a router or a part of the network. Throughput makes sure that, the device is not reduced to unacceptable levels.
- **Response Time** It is measured from the time a user requests a service to the time the service is granted.

**Security Management**

Security management is responsible for controlling access to the network based on the predefined policy.

**Accounting Management**

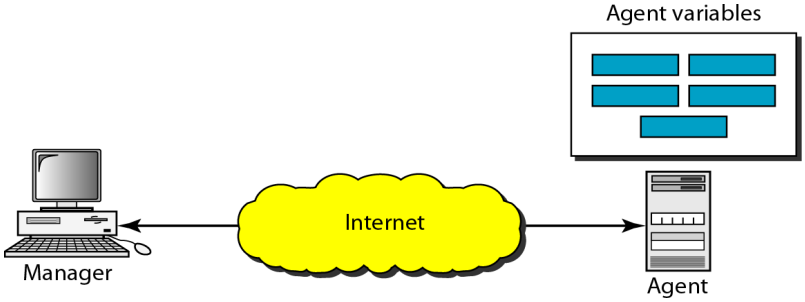
- Accounting management is the control of users' access to network resources through charges.
- Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network.

## SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

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

**Concept of SNMP**

- SNMP is an application level protocol that uses the concept of Manager and Agent.
- A manager controls and monitors a set ofagents.
- A manager may be a host and an Agent may be a router.
- SNMP can monitor devices made by different manufacturers and installed on different physical networks.
- SNMP can be used in LANs and WANs connected by routers.



**Managers and Agents**

- A Manager or a management station is a host that runs the SNMP client program.
- An Agent or a Managed station is a router or a host that runs the SNMP server program.

Management is achieved through simple interaction between a manager and an agent.

- Agent keeps performance information in a database.
- Manager has access to the values in the database.

Example: A router can store in variables such as the number of packets received and forwarded. The manager can fetch and compare the values of these two variables to see if the router is congested or not.

Management with SNMP is based on three basic ideas:

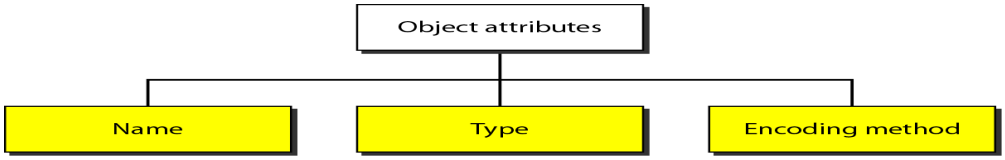
1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by sending warning message to the manager of an unusual situation.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

**Management Components**

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

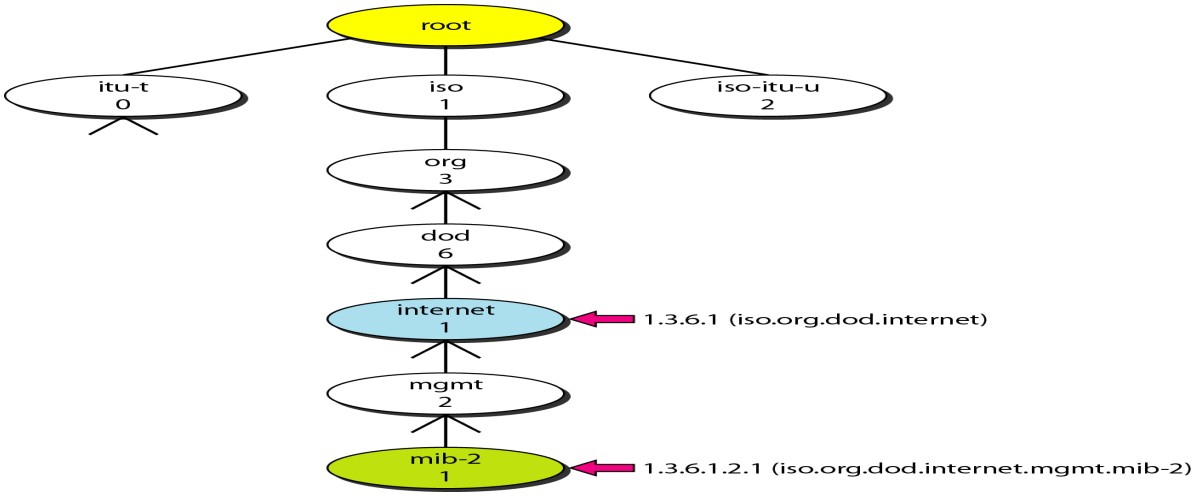
**Structure of Management Information (SMI)**

The Structure of Management Information version 2 (SMIv2) is a component for network management. Functions of SMI are:




**Name**

- SMI requires that each managed object (such as a router, a variable in a router, a value) have a unique name.
- To name objects globally SMI uses an object identifier, which is a hierarchical identifier based on a tree structure. The tree structure starts with an unnamed root.




- Each object can be defined by using a sequence of integers separated by dots.
- Tree structure can also define an object by using a sequence of textual names separated by dots.
- The integer-dot representation is used in SNMP. The name-dot notation is used by people.

  Example: The following shows the same object in two different notations:

  **iso.org.dod.internet.mgmt.mib-2 □1.3.6.1.2.1**

**Type**

- To define the data type, SMI uses fundamental Abstract Syntax Notation 1(ASN.l).
- SMI has two broad categories of data type: Simple and Structured.

**Simple Type**

The simple data types are atomic data types such a as Integer32, Octet String, IP address etc.

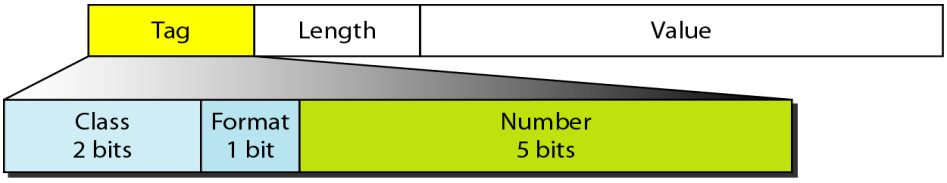
**Structured Type**

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

- A **Sequence** data type is a combination of simple data types, not necessarily of the same type. It is similar to the concept of a struct used in C programming.
- A **Sequence of** data type is a combination of simple data types all of the same type. It is similar to the concept of an array used in C-programming.

**Encoding Method**

SMI uses another standard Basic Encoding Rules (BER) to encode data to be transmitted over the network. BER specifies data to be encoded in following format: Tag, Length and Value.




**Tag**

The tag is a 1-byte field that defines the type of data. It is composed of three subfields:

- **Class (2 bits):** It defines the scope of the data.

  Four classes are defined: 00- universal, 01-Application wide, 10- context specific, 11- private.
- **Format** subfield indicates whether the data are simple (0) or structured(1).
- **Number** subfield further divides simple or structured data into subgroups.

  Example: In the universal class with simple format, INTEGER has a value of 2, OCTET STRING has a value of 4.

**Length**

- The length field is 1 or more bytes.
- If it is 1 byte, the most significant bit must be 0. Other 7 bits define the length of the data.
- If it is more than 1 byte, the most significant bit of the first byte must be 1. The other 7 bits of the first byte define the number of bytes needed to define the length.

**Value** field codes the value of the data according to the rules defined in BER.

**Example:** Show the following in encoding representation:

1. Define INTEGER14.

| 02 | 04 | 00 | 00 | 00 | 0E |
|---|---|---|---|---|---|
| 00000010 | 00000100 | 00000000 | 00000000 | 00000000 | 00001110 |
| Tag (integer) | Length (4 bytes) | Value (14) | | | |

2. Define OCTET STRING“HI”

| 04 | 02 | 48 | 49 |
|---|---|---|---|
| 00000100 | 00000010 | 01001000 | 01001001 |
| Tag (String) | Length (2 bytes) | Value (H) | Value (I) |

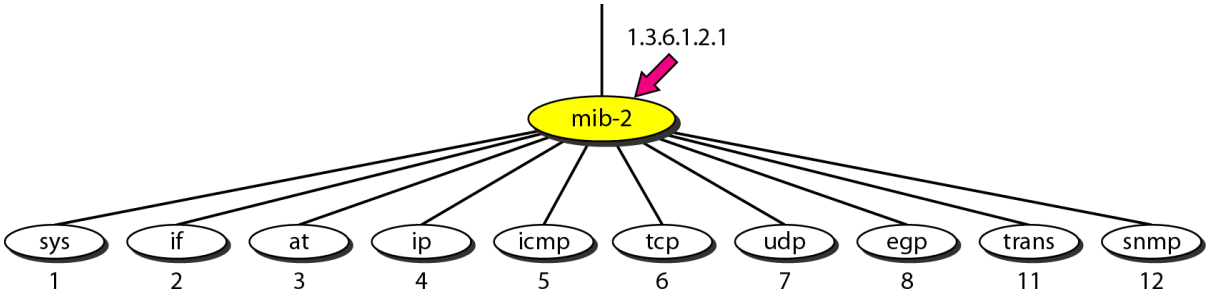
**Management Information Base(MIB2)**

**The basic purpose of MIB in SNMP is to provide a standardized framework for organizing, retrieving, and configuring data on network devices. This allows network administrators to effectively monitor and manage network performance and configurations across various devices.**

- MIB version 2 is the second component used in network management.
- MIB creates a collection of named objects, their types and their relationships to each other in an entity to be managed.
- Each agent has its own MIB2, which is a collection of all the objects that the manager can manage.
- The objects in MIB2 are categorized under 10 different groups: system,

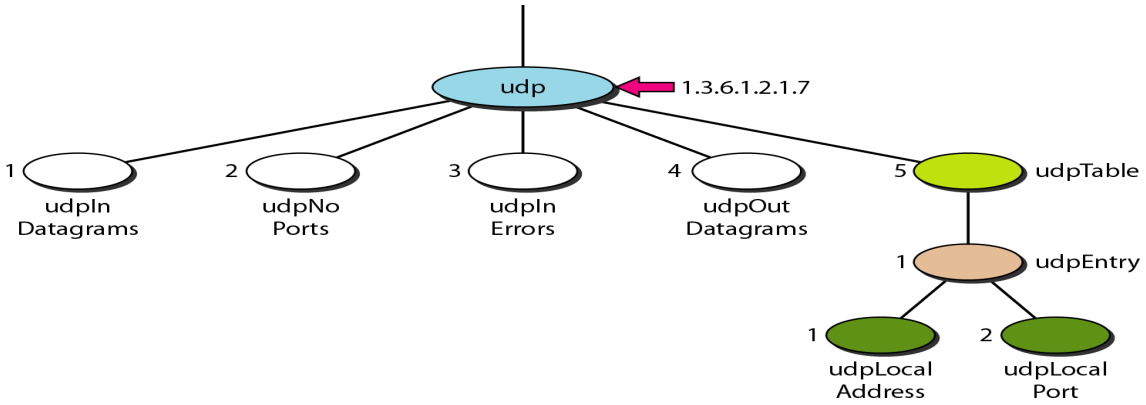
interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp.

- These groups are under the MIB-2 object in the object identifier tree.
- Each group has defined variables and tables.




**Accessing MIB Variables**

Let us take UDP group to show how to access different variables. To access any of the simple variables, we use the id of the group (1.3.6.1.2.1.7) followed by the id of the variable.




The following shows how to access each variable:

| | | |
|---|---|---|
| UdpIn Datagrams | □ | 1.3.6.1.2.1.7.1 |
| UdpNo Ports | □ | 1.3.6.1.2.1.7.2 |
| UdpIn Errors | □ | 1.3.6.1.2.1.7.3 |
| UdpOut Datagrams | □ | 1.3.6.1.2.1.7.4 |

The object identifiers define the variable not instances (contents). An instance suffix "**0**" should be added to show the instance of each variable.

| | | |
|---|---|---|
| udpInDatagrams.0 | □ | 1.3.6.1.2.1.7.1.0 |
| udpNoPorts.0 | □ | 1.3.6.1.2.1.7.2.0 |
| udpInErrors.0 | □ | 1.3.6.1.2.1.7.3.0 |
| udpOutDatagrams.0 | □ | 1.3.6.1.2.1.7.4.0 |

**Tables**

To identify a table, we first use the table id. To access the table, we have to define the table entries.

| | |
|---|---|
| Udp Table | □ 1.3.6.1.2.1.7.5 |
| udp Entry | □ 1.3.6.1.2.1.7.5.1 |

To access the entry we need to define each entity (field) in the entry.

Udp Local Address  1.3.6.1.2.1.7.5.1.1
Udp Local Port 1.3.6.1.2.1.7.5.1.2

- To access a specific instance (row) of the table, we add the index to the above ids. To access the instance of the local address for the first row, we use the identifier augmented with the instance index:

  udpLocalAddress.181.23.45.14.23  1.3.6.1.2.7.5.1.1.181.23.45.14.23

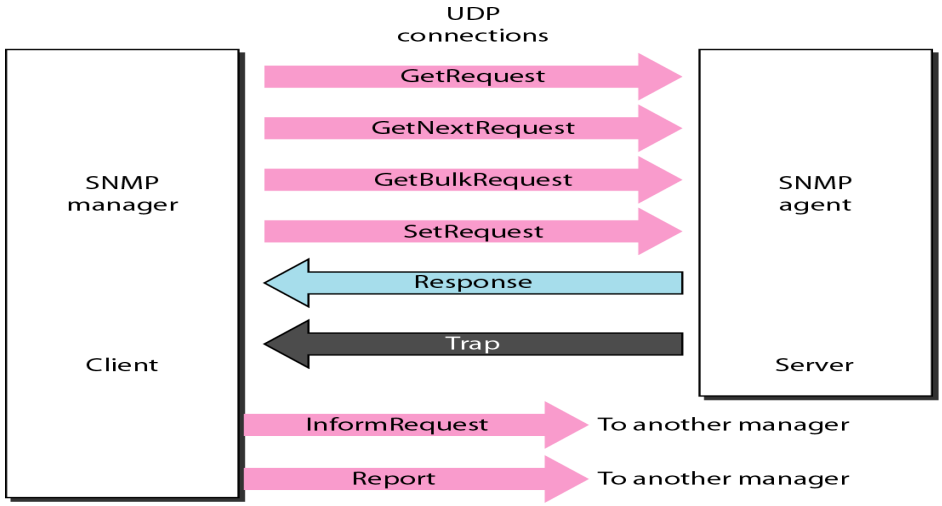
**Lexicographic Ordering**

- The object identifiers follow in lexicographic order.
- Tables are ordered column by column from the top to the bottom.
- The lexicographic ordering enables a manager to access a set of variables one after another by defining the first variable.

**SNMP version3 (SNMPv3)**

- SNMP defines the format of packets exchanged between a manager and an agent.
- SNMP interprets the result and creates statistics.
- The packets exchanged contain the object (variable) names and their status(values).
- SNMP is responsible for reading and changing these values.
- SNMP uses both SMI and MIB in Internet

network management. SNMPv3 defines eight types




of packets or PDUs.

1. **GetRequest** PDU is sent from the manager (client) to the agent (server) to retrieve the value of a variable or a set of variables.
2. **GetNextRequest** PDU is sent from the manager to the agent to retrieve the value of a variable. It is mostly used to retrieve the values of the entries in a table.
3. **GetBulkRequest** PDU is sent from the manager to the agent to retrieve a large amount of data.
4. **SetRequest** PDU is sent from the manager to the agent to set (store) a value in a variable.
5. **Response** PDU is sent from an agent to a manager in response to

GetRequest or GetNextRequest.

6. **Trap** PDU is sent from the agent to the manager to report an event. For example, if the agent is rebooted, it informs the manager and reports the time of rebooting.
7. **InformRequest** PDU is sent from one manager to another remote manager to get the value of some variables from agents under the control of the remote manager. The remote manager responds with a Response PDU.
8. **Report** PDU is designed to report some types of errors between managers. It is not yet in use.

# SOCKET PROGRAMMING

**Sockets:**

- A **socket** is one endpoint of a **two-way** communication link between two programs running on the network.
- The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication takes place.
- **Sockets** in computer networks are used for allowing the transmission of information between two processes of the same machines or different machines in the network.
- The socket is the combination of IP address and software port number used for communication between multiple processes.

Types of Sockets: There are two types: the **datagram** socket and the **stream** socket.

**Datagram Socket :** This is a type of network that has a connectionless point for sending and receiving packets. Bidirectional, can be in different order from sending sequence. Record boundaries are preserved, SOCKET_DGRAM

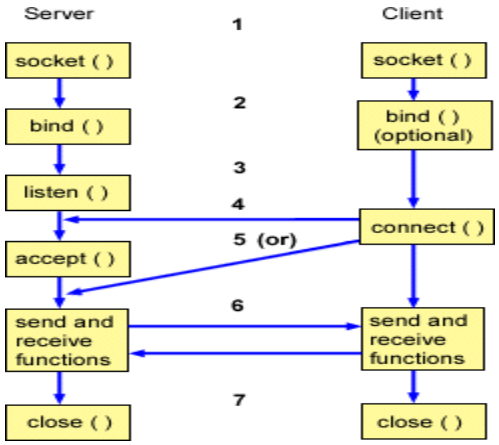
**Stream Socket** :A stream socket is a type of interprocess communications socket or network socket that provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well-defined mechanisms for creating and destroying connections and for detecting errors. Data can read from or write to these sockets as byte stream, SOCKET_STREAM

The socket API is a collection of socket calls that enables you to perform the following primary communication functions between application programs:

- Set up and establish connections to other users on the network
- Send and receive data to and from other users
- Close down connections

## How sockets work:

- A socket has a typical flow of events.
- The following figure shows the typical flow of events(and the sequence of issued APIs)for a connection-oriented socket session. An explanation of each event follows the figure.




| Function Call | Description |
|---|---|
| Socket() | To create a socket |
| Bind() | It's a socket identification like a telephone number to contact |
| Listen() | Ready to receive a connection |
| Connect() | Ready to act as a sender |
| Accept() | Confirmation, it is like accepting to receive a call from a sender |
| Write() | To send data |
| Read() | To receive data |
| Close() | To close a connection |

**This is a typical flow of events for a connection-oriented socket:**

1. The `socket()` API creates an endpoint for communications and returns a socket descriptor that represents the endpoint.
2. When an application has a socket descriptor, it can bind a unique name to the socket. Servers must bind a name to be accessible from the network.
3. The `listen()` API indicates a willingness to accept client connection requests. When a `listen()` API is issued for a socket, that socket cannot actively initiate connection requests. The `listen()` API is issued after a socket is allocated with a `socket()` API and the `bind()` API binds a name to the socket. A `listen()` API must be issued before an `accept()` API is issued.
4. The client application uses a `connect()` API on a stream socket to establish a connection to the server.
5. The server application uses the `accept()` API to accept a client connection request. The server must issue the `bind()` and `listen()` APIs successfully before it can issue an `accept()` API.
6. When a connection is established between stream sockets (between client and server), you can use any of the socket API data transfer APIs. Clients and servers have many data transfer APIs from which to choose, such as `send()`, `recv()`, `read()`, `write()`, and others.
7. When a server or client wants to stop operations, it issues a `close()` API to release any system resources acquired by the socket.

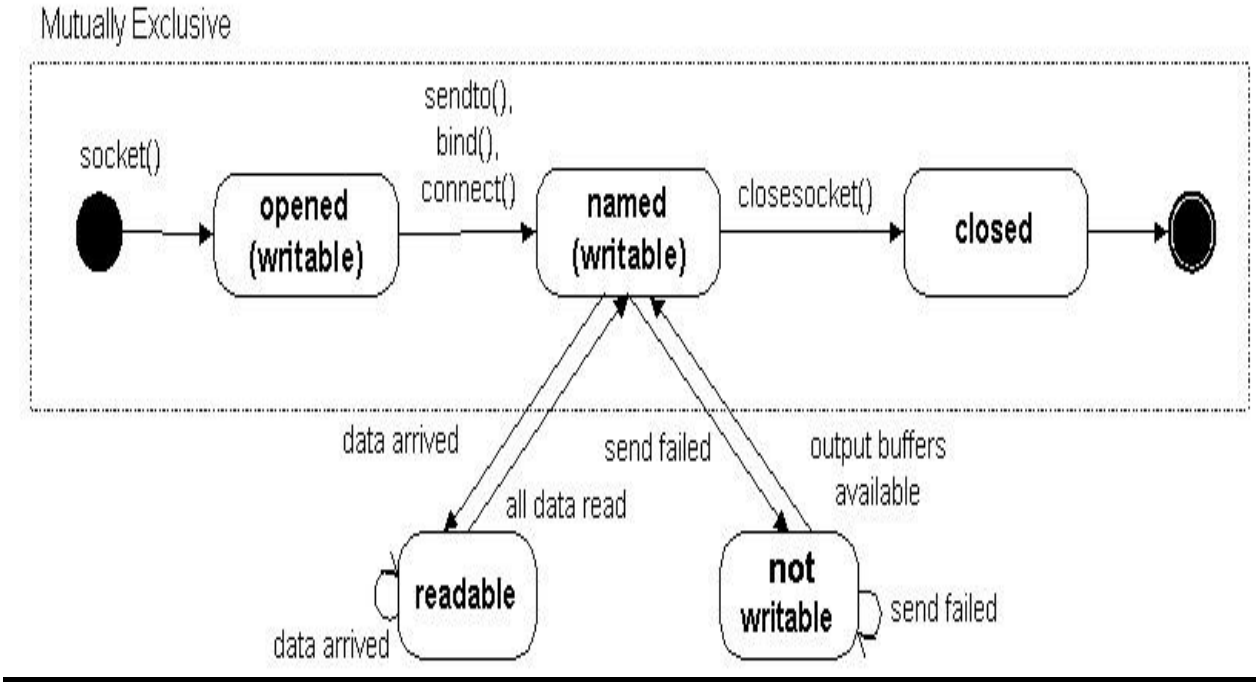
## Socket characteristics

- A socket is represented by an integer. That integer is called a *socket descriptor*.
- A socket exists as long as the process maintains an open link to the socket.
- You can name a socket and use it to communicate with other sockets in a communication domain.
- Sockets perform the communication when the server accepts connections from them, or when it exchanges messages with them.

## Socket States:

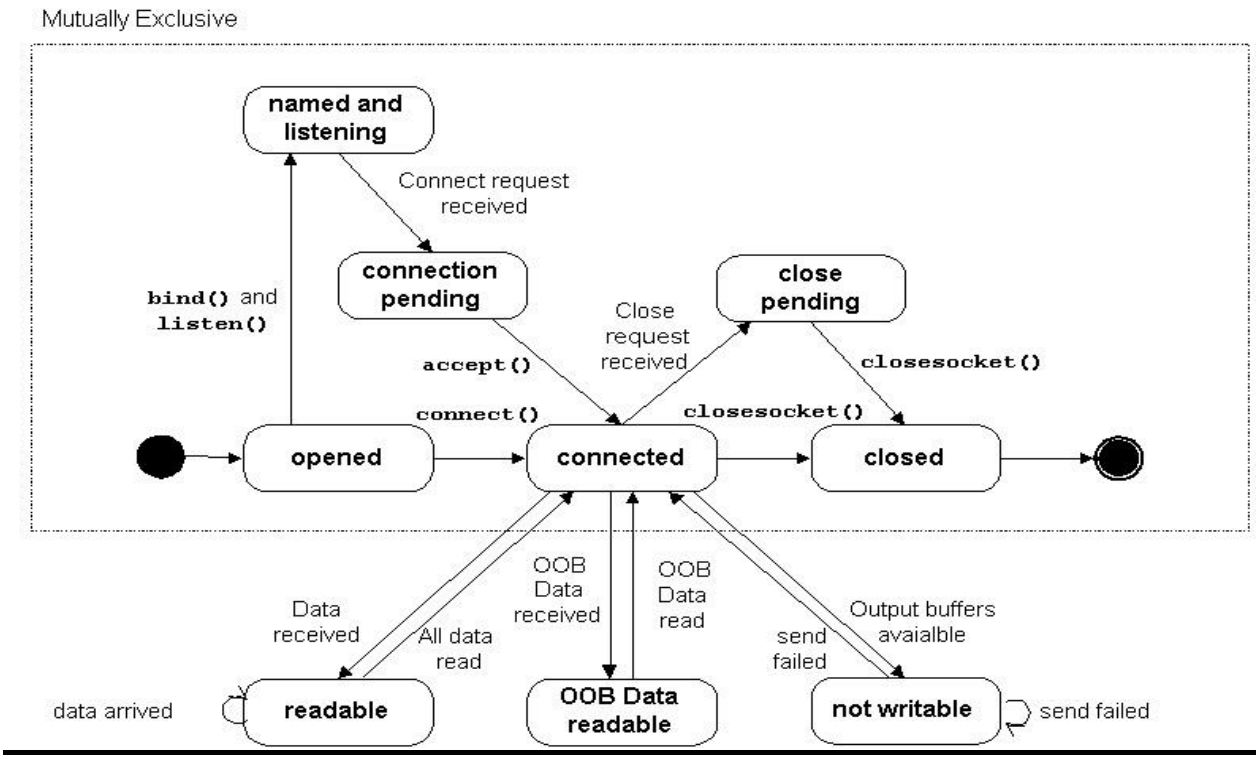
The state of a socket determines which network operations will succeed, which operations will block, and which operations with will fail (the socket state even determines the error code). Sockets have a finite number of states, and the WinSock API clearly defines the conditions that trigger a transition from one state to another.

**DATAGRAM SOCKET STATES*:***




| Socket State | Meaning |
|---|---|
| opened | **socket()** returned an unnamed socket (an unnamed socket is one that is not bound to a local address and port). The socket can be named explicitly with bind or implicitly with **sendto()** or **connect()**. |
| named | A named socket is one that is bound to a local address and a port. The socket can now send and/or receive. |
| readable | The network system received data and is ready to be read by the application using **recv()** or **recvfrom()**. |
| not writable | The network system does not have enough buffers to accommodate outgoing data. |
| closed | The socket handle is invalid. |

# STREAM SOCKET STATES:




| Socket State | Meaning |
|---|---|
| **opened** | **socket()** returned an unnamed socket (an unnamed socket is one that is not bound to a local address and port). The socket can be named explicitly with **bind()** or implicitly with **conenct()**. |
| **named and listening** | The socket is named (bound to a local address and port) and is ready to accept incoming connection requests. |
| **connection pending** | The network system received an incoming connection requests and is waiting for the application to respond. |
| **connected** | An association (virtual circuit) has been established between a local and remote host. Sending and receiving data is now possible |
| **readable** | The network system received data and is ready to be read by the application using **recv()** or **recvfrom()**. |
| **OOB readable** | Out Of Band data received by the network system received data and is ready to be read by the application (using **recv()** or **recvfrom()**.) |

| | |
|---|---|
| **Not writable** | The network system does not have enough buffers to accommodate outgoing data. |
| **close pending** | The virtual circuit is close. |
| **closed** | The socket handle is invalid. |

# Socket Options

Various types of options are available in a socket. There are various ways to get and set the options that affect a socket. They include,

- getsockopt and setsockopt functions
- fcntl function
- ioctl function

## (a) getsockopt and setsockopt

### Syntax

```
int getsockopt (int sockfd, int level, int optname, void *optval, socklen_t *optlen);

int setsockopt (int sockfd, int level, int optname, const void *optval, socklen_t optlen);

Return Value : Both return 0 on OK, -1 on error.
```

sockfd – refer to an open socket descriptor
level –specifies the code in the system that interprets the option. (i.e) general socket code or protocol specific code (IPv4, IPv6,TCP)
optname – name of the option
optval – It is a pointer to a variable from which the new value of the option is fetched by setsockopt or into which the current value of the option is stored by getsockopt.
Optlen – specifies the size of this variable.

## (b) fcntl

**fcntl** stands for "file control". This function performs various descriptor control operations.

### Syntax

```
int fcntl (int fd, int cmd, ….int arg);

Return value: 0 if OK, -1 on error
```

**(c) ioctl**

**ioctl** stands for "IO control".

Syntax

```
int ioctl (int fd, int request, ….void

*arg); Return value: 0 if OK, -1 on
```

# Generic Socket Options

These options are protocol independent options. These options are as follows.

**(1) SO_BROADCAST**

- This option enables or disables the ability of the process to send broadcast messages.
- Broadcasting is supported only for datagram sockets and only on networks that support the concept of broadcast message.
- An application must set this socket option before sending any broadcast message.
- If the destination address is a broadcast address and this socket option is not set, EACCESS is returned.

**(2) SO_DEBUG**

- This option is supported only by TCP.
- When this option is enabled for a TCP socket, the kernel keeps track of the detailed information about all the packets send and received by the TCP for the socket.
- These are kept in a circular buffer and can be examined with the trpt program.

**(3) SO_DONTROUTE**

- This option specifies that the outgoing packets are to bypass the normal routing mechanisms of the underlying protocol.
- According to the destination address given, the packets will be routed.
- So, a local interface will be identified and then the packet is routed.
- If the local interface cannot be identified, ENETUNREACH is returned.

- This option can also be applied to individual datagrams using MSG_DONTROUTE flag.
- This option is often used by the routing daemons to bypass the routing table and force a packet to be sent out a particular interface.

**(4) SO_ERROR**

- When an error occurs on a socket, the protocol module sets a variable named so_error for that socket to one of the standard unix Exxx values. This is called the pending error for the socket.
- This option can be fetched but cannot be set.
- The process can be notified about the error in one of the two ways.
    - If the process is blocked in a call to select on the socket, for either readability or writability.
    - If the process is using signal driven I/O, the SIGIO signal is generated for either the process or the process group.

**(5) SO_KEEPALIVE**

- The purpose of this option is to detect if the peer host crashes or become unreachable.
- When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours, TCP automatically sends a keep-alive probe to the peer.
- This probe is a TCP segment to which the peer must respond. One of the three scenarios result.
    - The peer responds with the expected ACK (If the peer is active)
    - The peer responds with an RST, which tells the local TCP that the peer host has been crashed and rebooted. So, the sockets pending error is set to ECONNRESET and the socket is closed.
    - There is no response from the peer to the keep-alive probe. TCP sends 8 additional probes, 75 seconds apart, trying to get a response from the peer. It will give up if there is no response within 11 minutes and 15 seconds from the time of sending the first probe. If there is no response, the sockets pending error is set to ETIMEDOUT and the socket are closed. If the peer host is unreachable, the pending error is set to EHOSTUNREACH.
- This option is normally used by servers, although clients can also use this option.

- Servers use this option because after establishment of connection, there may be a situation where the server may wait for the client request.
- But, if the client hosts crashes, powered off or connection drops, the server never knows about it and waits for the input that can never arrive. This is called a half open connection. The keep-alive option will detect these half open connections and terminate them.

**(6) SO_LINGER**

- This option specifies how the close function operates for a connection oriented protocol.
- By default, close returns immediately. But, if there is any data still remaining in the socket send buffer, the system will try to deliver the data to the peer.
- But, the SO_LINGER changes this default case.
- It requires the following structure to be passed between the user process and the kernel.

```
Struct linger
{
        int l_onoff;      /* 0 = off, non-zero = on
        */ int l_linger   /* linger time */
}
```

- When this socket option is set, any one of the following three scenarios takes place, depending on the values of the two structure members.
    - If l_onoff=0, the option is turned off. So, the value of l_linger is ignored and the TCP default applies (i.e) close returns immediately.
    - If l_onoff=nonzero and l_linger = 0, TCP aborts the connection when it is closed. (ie) TCP discards any data still remaining in the socket send buffer and sends a RST to its peer.
    - If l_onoff=nonzero and l_linger = nonzero, then the kernel will linger when the socket is closed. (i.e) If there is any data still remaining in the socket send buffer, the process is put to sleep until either,
        - All data is send and acknowledged by the peer TCP.
        - The linger time expires.
- Assume that the client writes data to the socket and then calls close. The following diagrams depict the various scenarios.

(a) Default situation

- By default, close returns immediately.

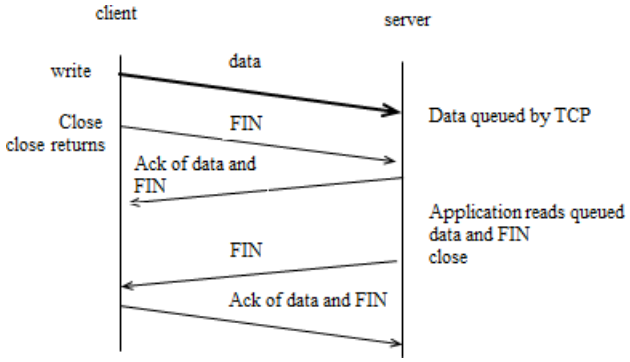



Fig 3.1 Default operation of close

We now need to look at exactly when *close* on a socket returns and what the actions and consequences are. In these cases, we assume that the client writes data to the socket and then calls close.

Fig. 3.1 shows the default scenario. Assume that when the client's data arrives, the server is temporarily busy, so the data is added to the socket receive buffer by its TCP. Similarly, the next segment, the client's FIN is also added.

But by default, the client's close returns immediately. As we see here, the client's close can return before the server reads the remaining data in its socket receive buffer. Therefore it is possible for the server host to crash before the server application reads this remaining data, and the client application will never know.

(b) SO_LINGER socket option is set and l_linger set to a positive value

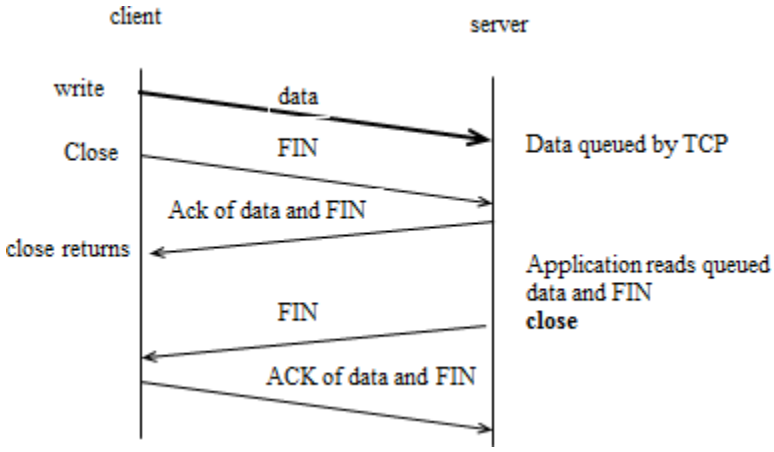



Fig 3.2 l_linger set to a positive value

In this scenario, the client sets the SO_LINGER option, specifying some positive linger time. When this occurs, the client's close does not return until all the client's data and its FIN have been acknowledged by the server TCP as shown in Fig 3.2.

The server host can crash before the server application reads its remaining data, and the client application will never know.

(c) SO_LINGER socket option set with l_linger set to small positive value

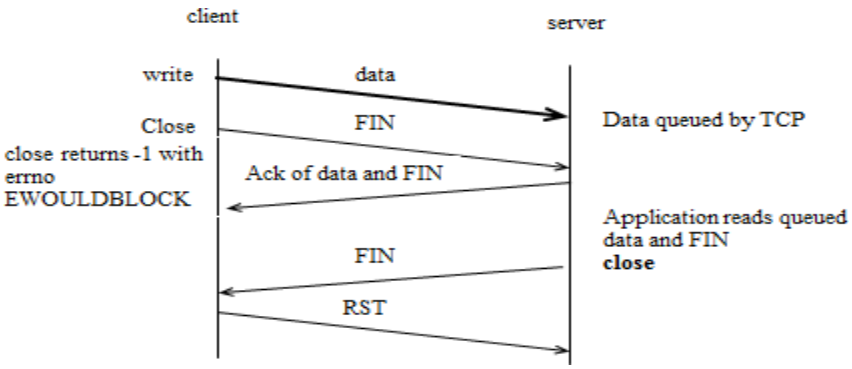



Fig 3.3 l_linger set to small positive value

Fig. 3.3 shows what can happen if the SO_LINGER option is set to a value that is too low. The basic principle here is that a successful return from close, with the SO_LINGER option set, only tells us that the data we sent (and our FIN) have been acknowledge by the peer TCP. It does not tell us whether the peer application has read the data. If we do not set the SO_LINGER option, we do not know whether the peer TCP has acknowledged the data.

(d) Using shutdown to show the peer has received the data

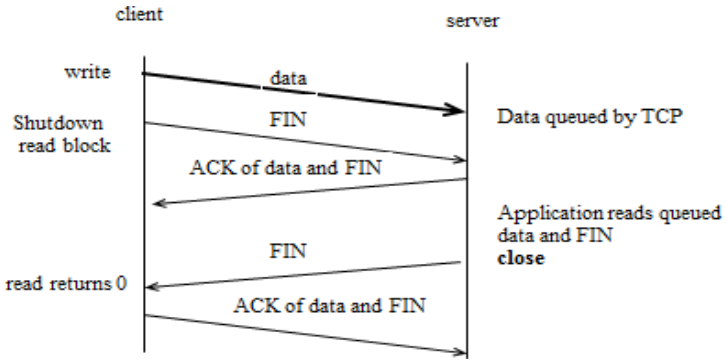



Fig. 3.4 Using shutdown

One way for the client to know that the server has read its data is to call shutdown (with SHUT_WR) instead of close and wait for the peer to close its end of the connection as shown in Fig. 3.4.

(e) Application ACK

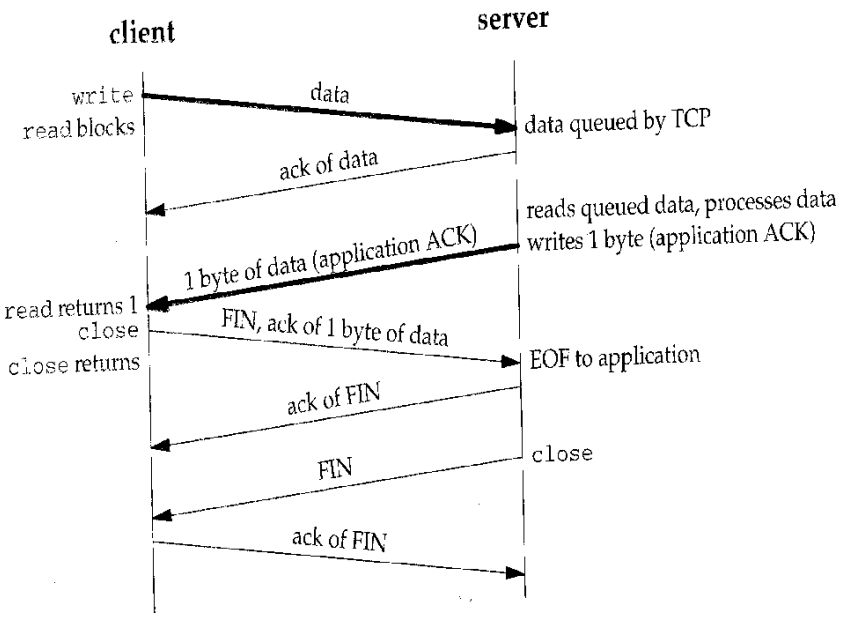



Fig. 3.5 Application ACK

Another way to know that the peer application has read our data is to use an application-level acknowledgment which requires coding in the server and client. In this case, the client waits for a 1 byte acknowledgment for each packet sent. Fig 3.5 shows the possible packet exchange.

**(7) SO_OOBINLINE**

- When this option is enabled, the out of band data will be placed in the normal input queue.
- When this occurs, the MSG_OOB flag to the receive functions cannot be used to read the out of band data.

(8) SO_RCVBUF and SO_SNDBUF

- Every socket has a send buffer and a receive buffer.
- Receive buffer - It is used to hold the received data until it is read by the application.
- Send buffer – It is used to hold the data to be send.

- The socket receive buffer has a limit in its window size. And, the peer can send data only upto the window size limit. The window size will be advertised to its peer while sending the SYN segment during connection establishment. This is TCPs flow control.
- If the peer ignores the advertised window and if it sends data beyond the window, the receiving TCP discards it.
- The default size of TCP send and receive buffers is 4096 bytes. But, newer systems use larger values from 8192 to 61440 bytes.
- The default size of UDP send buffer is 9000 bytes.
- The default size of UDP receive buffer is 40000 bytes.
- The main goal of this option is that these two options let us change the default sizes.

## (9) SO_RCVLOWAT and SO_SNDLOWAT

- Every socket has a receive low water mark and send low water mark.
- These are used by the select function.
- Receive low water mark – It is the amount of data that must be in the socket receive buffer for the select to return readable.
- Send low water mark – It is the amount of available space that must exist in the socket send buffer for select to return writable.
- Default receive low water mark is 1.
- Default send low water mark is 2048.
- These two socket options, let us change these two low water marks.

## (10) SO_RCVTIMEO and SO_SNDTIMEO

- These two socket options allow us to place a timeout on socket receive and send.
- This let us specify the timeout in seconds and microseconds.
- The timeout can be disabled by setting its value to 0 seconds and 0 microseconds.
- Both timeouts are disabled by default.
- The receive timeout affects the five input functions namely read, readv, recv, recvfrom and recvmsg.
- The send timeout affects the five output functions namely write, writev, send, sendto and sendmsg.

## (11) SO_REUSEADDR and SO_REUSEPORT

- SO_REUSEADDR serves four different purposes.
    - It allows a listening server to start and bind its well known port even if previously established connections exist that use this port as their local port. This condition is typically encountered as follows.
        - A listening server is started.
        - A connection request arrives and a child process is spawned to handle that client.
        - The listening server terminates, but the child continues to service the client on the existing connection.
        - The listening server is restarted.
    - It allows a new server to be started on the same port as an existing server that is bound to the wildcard address as long as each instance binds a different local IP address.
    - It allows a single process to bind the same port to multiple sockets, as long as each bind specifies a different local IP address.
    - It allows completely duplicate bindings : A bind of an IP address and port, when the same IP address and port are already bound to another socket, if the transport protocol supports it.
- This feature is supported only for UDP sockets.
- This feature is used with multicasting to allow the same application to be run multiple times on the same host.
- SO_REUSEADDR does the following.
    - It allows completely duplicate bindings, but only if each socket that wants to bind the same IP address and port specify this socket option.
    - It is considered equivalent to SO_REUSEPORT if the IP address being bound is a multicast address.
- Limitation : It is not supported by all systems.

## (12) SO_TYPE

- This option returns the socket type.
- The integer value returned is a value SOCK_STREAM or SOCK_DGRAM or SOCK_RAW.

## (13) SO_USELOOPBACK

- This option applies only to sockets in the routing domain.

- By default, this set to ON.
- When this option is enabled, the socket receives a copy of everything sent on the socket.

## IPv4 Socket Options

- These socket options are processed by IPv4. These options include the following.

(1) IP_HDRINCL

- If this option is set for a raw IP socket, we must build our own IP header for all the datagrams we send on the raw socket.
- Normally kernel builds the IP header for all datagrams, but some applications require to build their own IP header.
- When this option is set, we build a complete IP header, with the following exceptions.
  - IP always calculates and stores the IP header checksum.
  - If we set the IP identification field to 0, the kernel will set the field.
  - If the source IP address is INADDR_ANY, IP sets it to the primary IP address of the outgoing interface.
  - Setting IP options is implementation dependent.
  - Some fields must be in host byte order and some in network byte order. This is implementation dependent.

(2) IP_OPTIONS

- Setting this option allows us to set IP options in the IPv4 header.
- This requires intimate knowledge of the format of IP options in the IP header.

(3) IP_RECVDSTADDR

- This option causes the destination IP address of a received UDP datagram to be returned as ancillary data by recvmsg.

(4) IP_RECVIF

- This option causes the index of the interface on which a UDP datagram is received to be returned as ancillary data by recvmsg.

(5) IP_TOS

- This option let us set the type of service in the IP header for a TCP, UDP socket.
- TOS can be,
  - T – Throughput
  - R – Reliability

        - D – Delay
        - C - Cost

(6) IP_TTL

- TTL stands for Time to Live
- This option let us set and fetch the default TTL.

ICMPv6 Socket Option

- This socket option is processed by ICMPv6.

(1) ICMP_FILTER

- This option let us fetch and set an icmp6_filter structure that specifies which of the 256 possible ICMPv6 message types will be passed to the process on a raw socket.

## IPv6 Socket Option

- These socket options are processed by IPv6. These options include the following.

(1) IPv6_CHECKSUM

- This option specifies the byte offset into the user data where the checksum field is located.
- If this value is non-negative, the kernel will,
    - Compute and store a checksum for all outgoing packets.
    - Verify the received checksum on input, discarding packets with an invalid checksum.
- If the value is -1 (default), the kernel will not calculate and store the checksum for outgoing packets on this raw socket and will not verify the checksum for received packets.

(2) IPv6_DONTFRAG

- Setting this option disables the automatic insertion of a fragment header for UDP and raw sockets.
- When this option is set, output packets larger than Maximum Transfer Unit (MTU) of the outgoing interface will be dropped.

(3) IPv6_NEXTHOP

- This option specifies the next hop address for a datagram as a socket address structure and is a privileged operation.

(4) IPv6_PATHMTU

- This option cannot be set, only retrieved.
- When this option is retrieved, the current MTU as determined by PATH_MTU discovery is returned.

(5) IPv6_RECVDSTOPTS

- Setting this option specifies that any received IPv6 destination options are to be returned as ancillary data by recvmsg.

(6) IPv6_RECVHOPLIMIT

- Setting this option specifies that the received hop limit field is to be returned as ancillary data by recvmsg.

(7) IPv6_RECVHOPOPTS

- Setting this option specifies that any received IPv6 hop-by-hop options are to be returned as ancillary data by recvmsg.

(8) IPv6_RECVPATHMTU

- Setting this option specifies that the path MTU of a path is to be returned as ancillary data by recvmsg.

(9) IPv6_RECVPKTINFO

- Setting this option specifies that the following two pieces of information about a received IPv6 datagram are to be returned as ancillary data by recvmsg.
  - The destination IPv6 address
  - Arriving interface index

(10) IPv6_RECVRTHDR

- Setting this option specifies that a received IPv6 routing header is to be returned as an ancillary date by recvmsg.

(11) IPv6_RECVTCLASS

- Setting this option specifies that the received traffic class is to be returned as ancillary data by recvmsg.

(12) IPv6_UNICAST_HOPS

- Setting this option specifies the default hop limit for outgoing datagrams sent on the socket, while fetching the socket option returns the value of the hop limit that the kernel will use for the socket.

(13) IPv6_USE_MIN_MTU

- Setting this option avoids fragmentation.
  - When this option is set to 1, path MTU discovery is not performed and packets are sent using minimum MTU.

- When this option is set to 0, causes path MTU discovery to occur for all destinations.
- When this option is set to -1, path MTU discovery is performed.

(14) IPv6_V6ONLY

- Setting this option restricts it to IPv6 communication only.

(15) IPv6_XXX

- UDP socket uses, recvmsg and sendmsg
- TCP socket uses, getsockopt and setsockopt

# TCP Socket Options

(1) TCP_MAXSEG

- This socket option allows us to fetch or set the Maximum Segment Size (MSS) for a TCP connection.
- The value returned is the maximum amount of data that the TCP will send to the other end.
- The MSS is set while sending the SYN segment to the peer during connection establishment.
- The maximum amount of data that our TCP will send per segment can also change during the life of the connection if TCP supports path MTU discovery.
- If the route of the peer changes, this value will go up or down.

(2) TCP_NODELAY

- If this option is set, it disables TCP's Nagle algorithm.
- By default, this algorithm is enabled.
- Nagles algorithm avoids the syndrome caused in the sender side (i.e) if the sending side sends data too slowly, by sending each byte as a packet and waiting for the acknowledgment.

## Nagle's Algorithm

(1) It sends the first byte as it is as a packet and waits for an acknowledgment.

(2) When it receives the ACK, it does not send the further byte as it is, provided it waits until a certain number of bytes gets accumulated or till the ACK for the previous is arrived.

- The purpose of Nagle's algorithm is to reduce the number of small packets in WAN.
- Small packet is any packet smaller than MSS.
- The two common generators of small packets are the Rlogin and Telnet clients, since they send each keystroke as a separate packet.
- In a fast LAN, we normally donot notice a Nagle's algorithm because the time required for a small packet to be acknowledged is typically a few milliseconds, far less than the time between two successive characters that we type.
- But in a WAN, it takes nearly a second to acknowledge a small packet, so we can notice a delay in the character echoing and this delay is often exaggerated by the Nagle's algorithm.
- Consider the following example,
    - We type the six character string "hello!" with exactly 250 ms between each character.
    - The Round Trip Time (RTT) to the server is 600 ms and the server immediately sends back the echo of each character.
    - Assuming the Nagle's algorithm is disabled, we have the 12 packets as shown below.

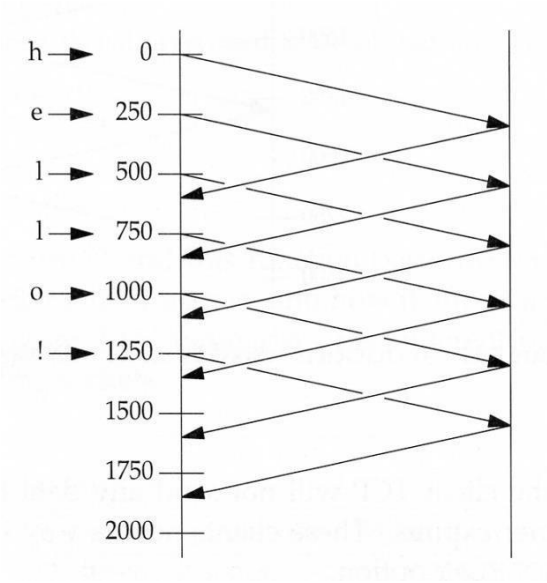



Fig 3.6 Nagle's algorithm (disabled)

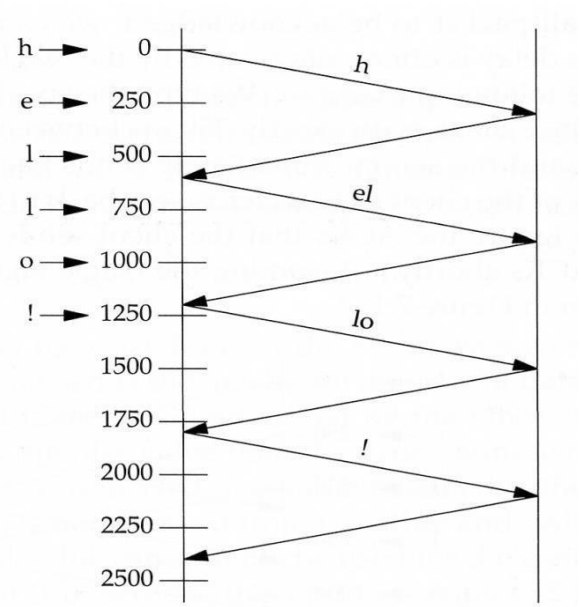



Fig. 3.7 Nagle's Algorithm (Enabled)

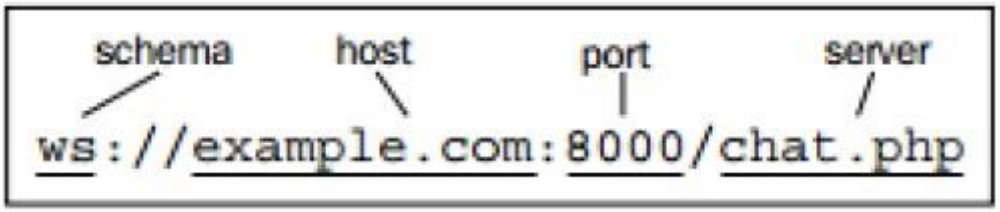
# Web sockets

Web sockets are defined as a two-way communication between the servers and the clients, which mean both the parties, communicate and exchange data at the same time.
This protocol defines a full duplex communication from the ground up. Web sockets take a step forward in bringing desktop rich functionalities to the web browsers.
The main features of web sockets are as follows:
- Web socket protocol is being standardized, which means real time communication between web servers and clients is possible with the help of this protocol.
- Web sockets are transforming to cross platform standard for real time communication between a client and the server.
- This standard enables new kind of the applications. Businesses for real time web application can speed up with the help of this technology.
- The biggest advantage of Web Socket is it provides a two-way communication (fullduplex) over a single TCP connection.

**URL**

HTTP has its own set of schemas such as http and https. Web socket protocol also has similar schema defined in its URL pattern.

The following image shows the Web Socket URL in tokens.




## Browser Support

The latest specification of Web Socket protocol is defined as **RFC 6455** – a proposed standard.

**RFC 6455** is supported by various browsers like Internet Explorer, Mozilla Firefox, Google Chrome, Safari, and Opera.

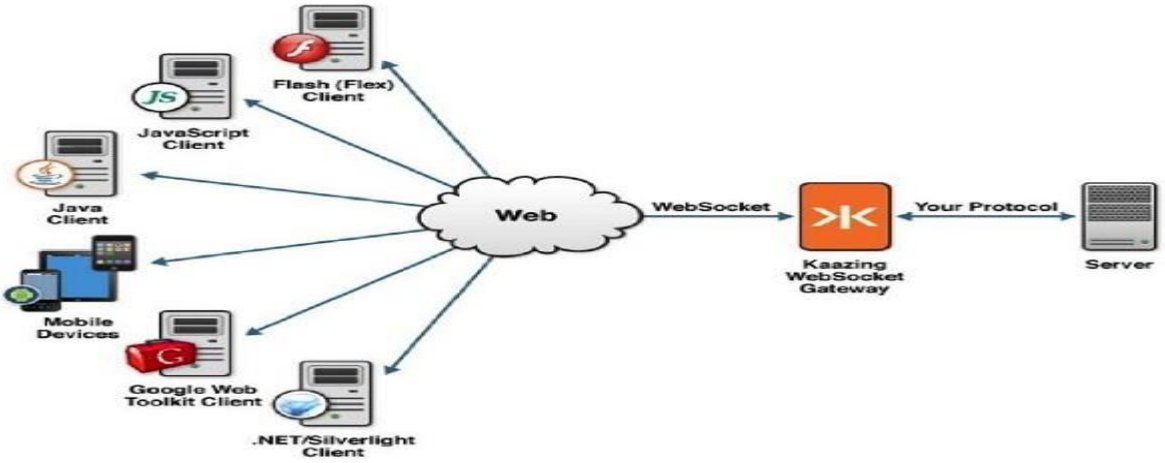
### Functionalities

Web Socket represents a major upgrade in the history of web communications. Before its existence, all communication between the web clients and the servers relied only on HTTP.

Web Socket helps in dynamic flow of the connections that are persistent full duplex. Full duplex refers to the communication from both the ends with considerable fast speed.

It is termed as a game changer because of its efficiency of overcoming all the drawbacks of existing protocols.

The following diagram describes the functionalities of Web Sockets −




Web Socket connections are initiated via HTTP; HTTP servers typically interpret Web Socket handshakes as an Upgrade request.

Web Sockets can both be a complementary add-on to an existing HTTP environment and can provide the required infrastructure to add web functionality. It relies on more advanced, full duplex protocols that allow data to flow in both directions between client and server.

## Events

There are four main Web Socket API **events** −

- Open
- Message
- Close
- Error

Each of the events are handled by implementing the functions like **onopen, onmessage**, **onclose** and **onerror** functions respectively. It can also be implemented with the help of addEventListener method.

The brief overview of the events and functions are described as follows −

### Open

Once the connection has been established between the client and the server, the open event is fired from Web Socket instance. It is called as the initial handshake between client and server. The event, which is raised once the connection is established, is called **onopen**.

### Message

Message event happens usually when the server sends some data. Messages sent by the server to the client can include plain text messages, binary data or images. Whenever the data is sent, the **onmessage** function is fired.

### Close

Close event marks the end of the communication between server and the client. Closing the connection is possible with the help of **onclose** event. After marking the end of communication with the help of **onclose** event, no messages can be further transferred between the server and the client. Closing the event can happen due to poor connectivity as well.

### Error

Error marks for some mistake, which happens during the communication. It is marked with the help of **onerror** event. **Onerror** is always followed by termination of connection. The detailed description of each and every event is discussed in further chapters.

## Actions

Events are usually triggered when something happens. On the other hand, actions are taken when a user wants something to happen. Actions are made by explicit calls using functions by users.

The Web Socket protocol supports two main actions, namely −

- send( )
- close( )

### send ( )

This action is usually preferred for some communication with the server, which includes sending messages, which includes text files, binary data or images.

A chat message, which is sent with the help of send() action, is as follows −

```
// get text view and button for submitting the message
vartextsend=document.getElementById("text-view");
varsubmitMsg=document.getElementById("tsend-button");

//Handling the click event
submitMsg.onclick=function(){
// Send the data
socket.send(textsend.value);
}
```

**Note** − Sending the messages is only possible if the connection is open.

### close ( )

This method stands for goodbye handshake. It terminates the connection completely and no data can be transferred until the connection is re-established.

```
vartextsend=document.getElementById("text-view");
varbuttonStop=document.getElementById("stop-button");

//Handling the click event
buttonStop.onclick=function(){
// Close the connection if open
if(socket.readyState===WebSocket.OPEN){
socket.close();
}
}
```

It is also possible to close the connection deliberately with the help of following code snippet −

```
socket.close(1000,”DeliberateConnection”);
```

Opening Connections

Once a connection has been established between the client and the server, the open event is fired from Web Socket instance. It is called as the initial handshake between client and server.

The event, which is raised once the connection is established, is called the onopen. Creating Web Socket connections is really simple. All you have to do is call the WebSocket constructor and pass in the URL of your server.

The following code is used to create a Web Socket connection −

```
// Create a new WebSocket.
var socket =newWebSocket('ws://echo.websocket.org');
```

Once the connection has been established, the open event will be fired on your Web Socket instance.

onopen refers to the initial handshake between client and the server which has lead to the first deal and the web application is ready to transmit the data.

The following code snippet describes opening the connection of Web Socket protocol −

```
socket.onopen=function(event){
console.log(“Connection established”);
// Display user friendly messages for the successful establishment of connection
var.label=document.getElementById(“status”);
label.innerHTML=”Connection established”;
}
```

It is a good practice to provide appropriate feedback to the users waiting for the Web Socket connection to be established. However, it is always noted that Web Socket connections are comparatively fast.

**Handling Errors**

Once a connection has been established between the client and the server, an **open** event is fired from the Web Socket instance. Error are generated for mistakes, which take place during the communication. It is marked with the help of **onerror** event. **Onerror** is always followed by termination of connection.

The **onerror** event is fired when something wrong occurs between the communications. The event **onerror** is followed by a connection termination, which is a **close** event.

A good practice is to always inform the user about the unexpected error and try to reconnect them.

```
socket.onclose=function(event){
console.log("Error occurred.");

// Inform the user about the error.
var label =document.getElementById("status-label");
label.innerHTML="Error: "+event;
}
```

When it comes to error handling, you have to consider both internal and external parameters.

- Internal parameters include errors that can be generated because of the bugs in your code, or unexpected user behavior.
- External errors have nothing to do with the application; rather, they are related to parameters, which cannot be controlled. The most important one is the network connectivity.
- Any interactive bidirectional web application requires, well, an active Internet connection.

## Send & Receive Messages

The **Message** event takes place usually when the server sends some data. Messages sent by the server to the client can include plain text messages, binary data, or images. Whenever data is sent, the **onmessage** function is fired.

This event acts as a client's ear to the server. Whenever the server sends data, the **onmessage** event gets fired.

The following code snippet describes opening the connection of Web Socket protocol.

```
connection.onmessage=function(e){
varserver_message=e.data;
   console.log(server_message);
}
```

It is also necessary to take into account what kinds of data can be transferred with the help of Web Sockets. Web socket protocol supports text and binary data. In terms of Javascript, **text** refers to as a string, while binary data is represented like **ArrayBuffer**.

Web sockets support only one binary format at a time. The declaration of binary data is done explicitly as follows −

```
socket.binaryType=”arrayBuffer”;
socket.binaryType=”blob”;
```

### Strings

Strings are considered to be useful, dealing with human readable formats such as XML and JSON. Whenever **onmessage** event is raised, client needs to check the data type and act accordingly.

The code snippet for determining the data type as String is mentioned below −

```
socket.onmessage=function(event){

if(typeOfevent.data===String){
console.log(“Received data string”);
}
}
```

## Closing a Connection

**Close** event marks the end of a communication between the server and the client. Closing a connection is possible with the help of **onclose** event. After marking the end of communication

with the help of **onclose** event, no messages can be further transferred between the server and the client. Closing the event can occur due to poor connectivity as well.

The **close()** method stands for **goodbye handshake**. It terminates the connection and no data can be exchanged unless the connection opens again.

Similar to the previous example, we call the **close()** method when the user clicks on the second button.

```
vartextView=document.getElementById("text-view");
varbuttonStop=document.getElementById("stop-button");

buttonStop.onclick=function(){
// Close the connection, if open.
if(socket.readyState===WebSocket.OPEN){
socket.close();
}
}
```

It is also possible to pass the code and reason parameters we mentioned earlier as shown below.

socket.close(1000, "Deliberate disconnection");

# API

API, an abbreviation of Application Program Interface, is a set of routines, protocols, andtools for building software applications.

Some important features are −

- The API specifies how software components should interact and APIs should be usedwhen programming graphical user interface (GUI) components.
- A good API makes it easier to develop a program by providing all the building blocks.
- REST, which typically runs over HTTP is often used in mobile applications, socialwebsites, mashup tools, and automated business processes.
- The REST style emphasizes that interactions between the clients and services is enhancedby having a limited number of operations (verbs).
- Flexibility is provided by assigning resources; their own unique Universal ResourceIdentifiers (URIs).
- REST avoids ambiguity because each verb has a specific meaning (GET, POST, PUT andDELETE)

### Advantages of Web Socket

Web Socket solves a few issues with REST, or HTTP in general −

### Bidirectional

HTTP is a unidirectional protocol where the client always initiates a request. The server processes and returns a response, and then the client consumes it. Web Socket is a bi-directional protocol where there are no predefined message patterns such as request/response. Either the client or the server can send a message to the other party.

### Full Duplex

HTTP allows the request message to go from the client to the server and then the server sendsa response message to the client. At a given time, either the client is talking to the server or theserver is talking to the client. Web Socket allows the client and the server to talk independent of each other.

### Single TCP Connection

Typically, a new TCP connection is initiated for an HTTP request and terminated after the response is received. A new TCP connection needs to be established for another HTTP request/response. For Web Socket, the HTTP connection is upgraded using standard HTTP upgrade mechanism and the client and the server communicate over that same TCP connectionfor the lifecycle of Web Socket connection.

## Communicating with Server

The Web has been largely built around the request/response paradigm of HTTP. A client loads up a web page and then nothing happens until the user clicks onto the next page. Around 2005, AJAX started to make the web feel more dynamic. Still, all HTTP communication is steered by the client, which requires user interaction or periodic polling to load new data from the server.

Technologies that enable the server to send the data to a client in the very moment when it knows that new data is available have been around for quite some time. They go by names such as **"Push"** or **“Comet”**.

With **long polling**, the client opens an HTTP connection to the server, which keeps it open until sending response. Whenever the server actually has new data, it sends the response. Long polling and the other techniques work quite well. However, all of these share one problem, they carry the overhead of HTTP, which does not make them well suited for low latency applications. For example, a multiplayer shooter game in the browser or any other online game with a real-time component.

**Bringing Sockets to the Web**

The Web Socket specification defines an API establishing "socket" connections between a web browser and a server. In layman terms, there is a persistent connection between the client and the server and both parties can start sending data at any time.

Web socket connection can be simply opened using a constructor −

var connection = new WebSocket('ws://html5rocks.websocket.org/echo', ['soap', 'xmpp']);

**ws** is the new URL schema for WebSocket connections. There is also **wss**, for secure WebSocket connection the same way **https** is used for secure HTTP connections.

Attaching some event handlers immediately to the connection allows you to know when the connection is opened, received incoming messages, or there is an error.

The second argument accepts optional **subprotocols**. It can be a string or an array of strings. Each string should represent a **subprotocol** name and server accepts only one of passed **subprotocols** in the array. Accepted **subprotocol** can be determined by accessing protocol property of WebSocket object.

```
// When the connection is open, send some data to the server
connection.onopen=function(){
connection.send('Ping');// Send the message 'Ping' to the server
};

// Log errors
connection.onerror=function(error){
console.log('WebSocket Error '+ error);
};

// Log messages from the server
connection.onmessage=function(e){
console.log('Server: '+e.data);
};
```

As soon as we have a connection to the server (when the open event is fired) we can start sending data to the server using the send (your message) method on the connection object. It used to support only strings, but in the latest specification, it now can send binary messages too. To send binary data, Blob or ArrayBuffer object is used.

```
// Sending String
connection.send('your message');
```

```
// Sending canvas ImageData as ArrayBuffer
varimg=canvas_context.getImageData(0,0,400,320);
var binary =newUint8Array(img.data.length);

for(vari=0;i<img.data.length;i++){
    binary[i]=img.data[i];
}

connection.send(binary.buffer);

// Sending file as Blob
var file =document.querySelector('input[type = "file"]').files[0];
connection.send(file);
```

Equally, the server might send us messages at any time. Whenever this happens the onmessage callback fires. The callback receives an event object and the actual message is accessible via the `data` property.

WebSocket can also receive binary messages in the latest spec. Binary frames can be received in Blob or ArrayBuffer format. To specify the format of the received binary, set the binaryType property of WebSocket object to either 'blob' or 'arraybuffer'. The default format is 'blob'.

```
// Setting binaryType to accept received binary as either 'blob' or 'arraybuffer'
connection.binaryType='arraybuffer';
connection.onmessage=function(e){
    console.log(e.data.byteLength);// ArrayBuffer object if binary
};
```

Another newly added feature of WebSocket is extensions. Using extensions, it will be possible to send frames compressed, multiplexed, etc.

```
// Determining accepted extensions
console.log(connection.extensions);
```

## Cross-Origin Communication

Being a modern protocol, cross-origin communication is baked right into WebSocket. WebSocket enables communication between parties on any domain. The server decides whether to make its service available to all clients or only those that reside on a set of well-defined domains.

## Proxy Servers

Every new technology comes with a new set of problems. In the case of WebSocket it is the compatibility with proxy servers, which mediate HTTP connections in most company networks. The WebSocket protocol uses the HTTP upgrade system (which is normally used for HTTP/SSL) to "upgrade" an HTTP connection to a WebSocket connection. Some proxy servers do not like this and will drop the connection. Thus, even if a given client uses the WebSocket protocol, it may not be possible to establish a connection. This makes the next section even more important :)

## The Server Side

Using WebSocket creates a whole new usage pattern for server side applications. While traditional server stacks such as LAMP are designed around the HTTP request/response cycle they often do not deal well with a large number of open WebSocket connections. Keeping a large number of connections open at the same time requires an architecture that receives high concurrency at a low performance cost.

# WebSockets - Security

Protocol should be designed for security reasons. WebSocket is a brand-new protocol and not all web browsers implement it correctly. For example, some of them still allow the mix of HTTP and WS, although the specification implies the opposite. In this chapter, we will discuss a few common security attacks that a user should be aware of.

## Denial of Service

Denial of Service (DoS) attacks attempt to make a machine or network resource unavailable to the users that request it. Suppose someone makes an infinite number of requests to a web server with no or tiny time intervals. The server is not able to handle each connection and will either stop responding or will keep responding too slowly. This can be termed as Denial of service attack.

Denial of service is very frustrating for the end users, who could not even load a web page.

DoS attack can even apply on peer-to-peer communications, forcing the clients of a P2P network to concurrently connect to the victim web server.

## Man-in-the-middle

Suppose a person **A** is chatting with his friend **B** via an IM client. Some third person wants to view the messages you exchange. So, he makes an independent connections with both the persons. He also sends messages to person **A** and his friend **B**, as an invisible intermediate to your communication. This is known as a man-in-the-middle attack.

The man-in-the-middle kind of attack is easier for unencrypted connections, as the intruder can read the packages directly. When the connection is encrypted, the information has to be decrypted by the attacker, which might be way too difficult.

From a technical aspect, the attacker intercepts a public-key message exchange and sends the message while replacing the requested key with his own. Obviously, a solid strategy to make the attacker's job difficult is to use SSH with WebSockets.

Mostly when exchanging critical data, prefer the WSS secure connection instead of the unencrypted WS.

### XSS

Cross-site scripting (XSS) is a vulnerability that enables attackers to inject client-side scripts into web pages or applications. An attacker can send HTML or Javascript code using your application hubs and let this code be executed on the clients' machines.

# WebSocket Native Defense Mechanisms

By default, the WebSocket protocol is designed to be secure. In the real world, the user might encounter various issues that might occur due to poor browser implementation. As time goes by, browser vendors fix any issues immediately.

An extra layer of security is added when secure WebSocket connection over SSH (or TLS) is used.

In the WebSocket world, the main concern is about the performance of a secure connection. Although there is still an extra TLS layer on top, the protocol itself contains optimizations for this kind of use, furthermore, WSS works more sleekly through proxies.

## Client-to-Server masking

Every message transmitted between a WebSocket server and a WebSocket client contains a specific key, named masking key, which allows any WebSocket-compliant intermediaries to unmask and inspect the message. If the intermediary is not WebSocket-compliant, then the message cannot be affected. The browser that implements the WebSocket protocol handles masking.

## Security Toolbox

Finally, useful tools can be presented to investigate the flow of information between your WebSocket clients and server, analyze the exchanged data, and identify possible risks.

## Browser Developer Tools

Chrome, Firefox, and Opera are great browsers in terms of developer support. Their built-in tools help us determine almost any aspect of client-side interactions and resources. It plays a great role for security purposes.