UNIT-1
Computer Networks and the Internet

1. What Is the Internet?

The Internet is a computer network that interconnects hundreds of millions of computing devices throughout the world.

A Nuts-and-Bolts Description

Computer networking is a collection of computers and other devices that can exchange data and share resources with each other.

The devices are called hosts or end systems, devices were primarily traditional desktop PCs, workstations, servers, smart phones, laptops etc.,

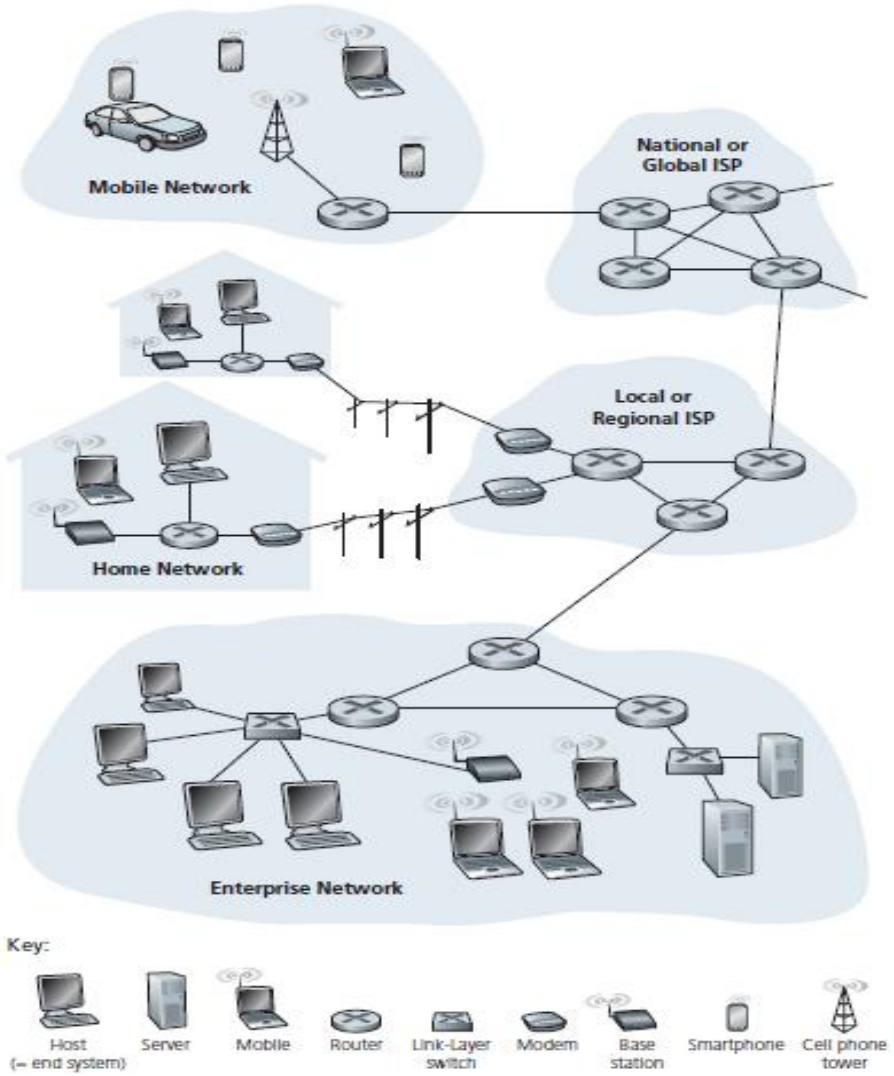


Fig: Some pieces of the Internet

- End systems are connected together by a network of **communication links** and **packet switches**.
- there are many types of communication links, which are made up of different types of physical media, including coaxial cable, copper wire, optical fiber, and radio spectrum.
- Different links can transmit data at different rates, with the **transmission rate** of a

link measured in bits/second.

- When one end system has data to send to another end system, the sending end system segments the data and adds header bytes to each segment. The resulting packages of information, known as **packets.**
- A packet switch takes a packet arriving on one of its incoming communication links and forwards that packet on one of its outgoing communication links. The two most prominent types in today's Internet are **routers** and **link-layer switches**. Both types of switches forward packets toward their ultimate destinations.
- The sequence of communication links and packet switches traversed by a packet from the sending end system to the receiving end system is known as a **route** or **path.**
- End systems access the Internet through **Internet Service Providers (ISPs),**including residential ISPs such as local cable or telephone companies; corporate ISPs; university ISPs; and ISPs that provide WiFi access in airports, hotels, coffee shops, and other public places.
- End systems, packet switches, and other pieces of the Internet run **protocols** that control the sending and receiving of information within the Internet. The **Transmission Control Protocol (TCP)** and the **Internet Protocol (IP)** are two of the most important protocols in the Internet.
- The IP protocol specifies the format of the packets that are sent and received among routers and end systems. The Internet's principal protocols are collectively known as **TCP/IP**.

A Services Description

- As *an infrastructure that provides services to applications*. These applications include electronic mail, Web surfing, social networks, instant messaging, video streaming, distributed games, peer-to-peer (P2P) file sharing, television over the Internet, remote login, and much, much more.
- The applications are said to be **distributed applications**, since they involve multiple end systems that exchange data with each other. Importantly, Internet applications run on end systems—they do not run in the packet switches in the network core.
- End systems attached to the Internet provide an **Application Programming Interface (API)** that specifies how a program running on one end system asks the Internet infrastructure to deliver data to a specific destination program running on another end system.
- This Internet API is a set of rules that the sending program must follow so that the Internet can deliver the data to the destination program.

What Is a Protocol?

A **protocol** defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.

A Human Analogy and Network Protocols

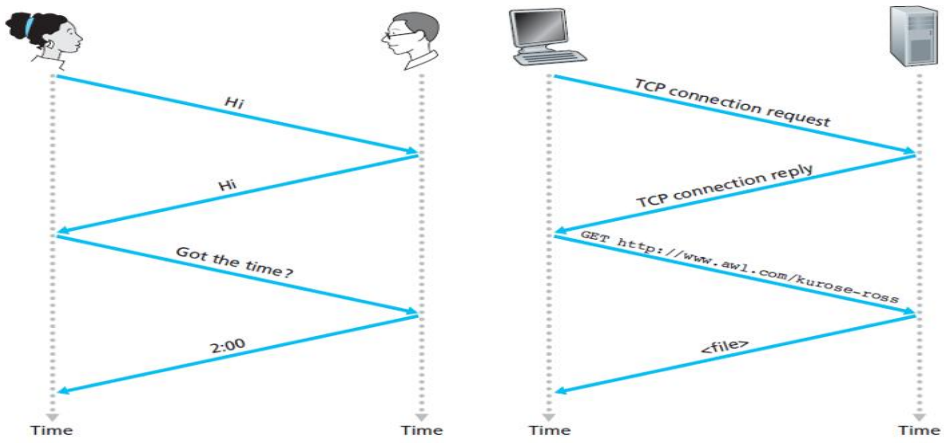


Fig: A human protocol and a computer network protocol

Human protocol

- Entities exchanging messages
- There are specific messages we send, and specific actions we take in response to the received reply messages or other event.

Computer Network Protocols

- Machines rather than humans.
- All communication activity in internet governed by protocols.

 Protocols in routers determine a packet's path from source to destination. Protocols are running everywhere in the Internet. The Internet and computer network make extensive use of protocols. Different protocols are used to accomplish different communication tasks.

2. The Network Edge

- The network edge refers to the area where a device or local network interfaces with the internet. The edge is close to the devices it is communicating with and is the entry point to the network.
- The network edge refers to endpoints. It is the first step between endpoints and the core of the network.
- The internet end systems include desktop computers, servers (web and email servers) and mobile computers (laptops, smart phones, tablets).

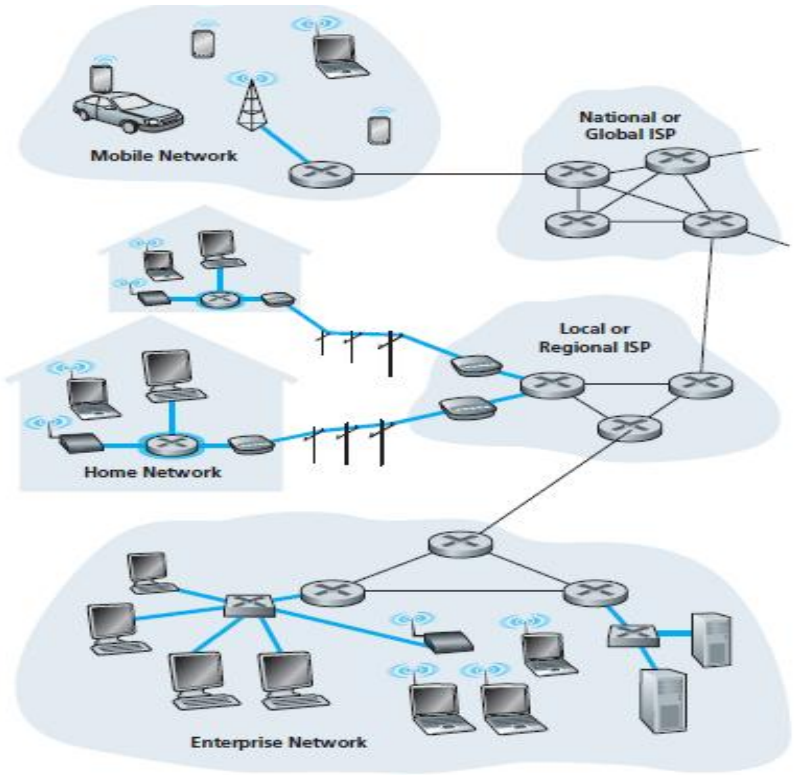


Fig: Access Networks

- End systems are also referred to as *hosts* because they run application programs such as a Web browser program, a Web server program, an e-mail client program, or an e-mail server program.
- Hosts are sometimes further divided into two categories: **clients** and **servers**.

Access Networks

The network that physically connects an end system to the first router (also known as the "edge router") on a path from the end system to any other distant end system.

Home Access: DSL, Cable, FTTH, Dial-Up, and Satellite

i) *DSL (digital subscriber line):* Today, the two most prevalent types of broadband residential access are **digital subscriber line (DSL)** and cable.

- DSL is a wire line transmission technology that transmits data faster.
- A residence typically obtains DSL Internet access from the same local telephone company (telco) that provides its wired local phone access. When DSL is used, a customer's telco is also its ISP.

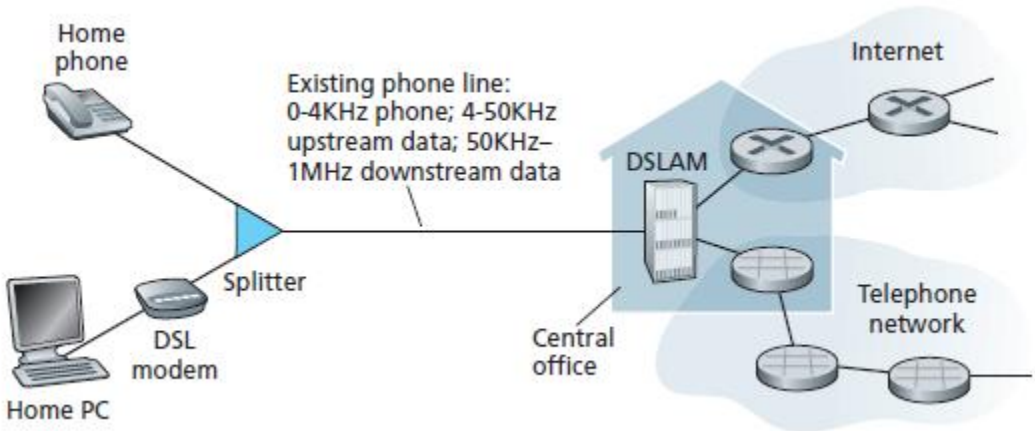


Fig: DSL Internet access

- As shown in Figure, each customer's DSL modem uses the existing telephone line to exchange data with a digital subscriber line access multiplexer (DSLAM) located in the telco's local central office (CO).
- The home's DSL modem takes digital data and translates it to high frequency tones for transmission over telephone wires to the CO; the analog signals from many such houses are translated back into digital format at the DSLAM.
- The residential telephone line carries both data and traditional telephone signals simultaneously.
- On the customer side, a splitter separates the data and telephone signals arriving to the home and forwards the data signal to the DSL modem.
- On the telco side, in the CO, the DSLAM separates the data and phone signals and sends the data into the Internet. Hundreds or even thousands of households connect to a single DSLAM.

ii) Cable: While DSL makes use of the telco's existing local telephone infrastructure, **cable Internet access** makes use of the cable television company's existing cable television infrastructure.
A residence obtains cable Internet access from the same company that provides its cable television.

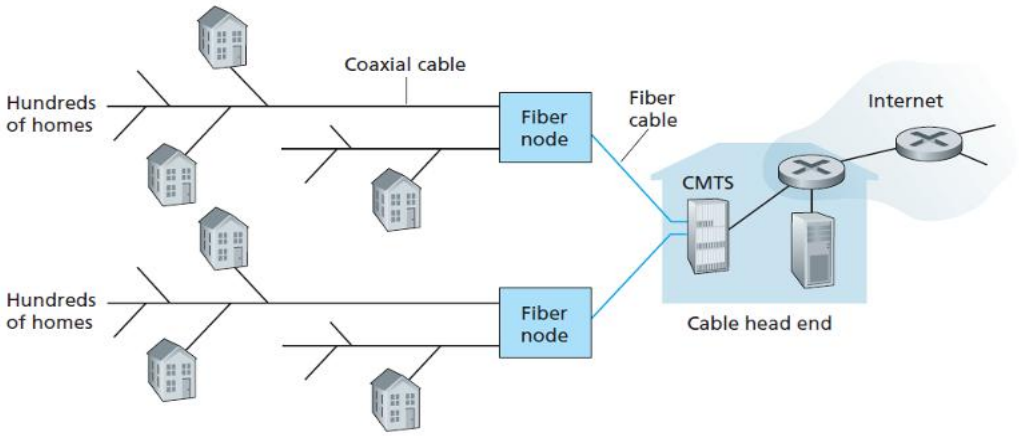


Fig: A hybrid fiber-coaxial access network

- As shows in Figure, fiber optics connects the cable head end to neighborhood-level junctions, from which traditional coaxial cable is then used to reach individual houses and apartments.
- Because both fiber and coaxial cable are employed in this system, it is often referred to as hybrid fiber coax **(HFC).**
- Cable internet access requires special modems, called cable modems. The cable modem is typically an external device and connects to the home PC through an Ethernet port.
- At the cable head end, the cable modem termination system (CMTS) serves a similar function as the DSL network's DSLAM—turning the analog signal sent from the cable modems in many downstream homes back into digital format.
- Cable modems divide the HFC network into two channels, a downstream and an upstream channel.
- One important characteristic of cable Internet access is that it is a shared broadcast medium. In particular, every packet sent by the head end travels downstream on every link to every home and every packet sent by a home travels on the upstream channel to the head end.

iii) FTTH (Fiber To The Home): FTTH includes fiber-optic access solutions designed for residential deployments. In FTTH networks, fibers are directly connected to individual homes or buildings.

- The FTTH can provide an optical fiber path from the CO directly to the home.
- There are two competing optical-distribution network architectures that perform this splitting: active optical networks (AONs) and passive optical networks (PONs).

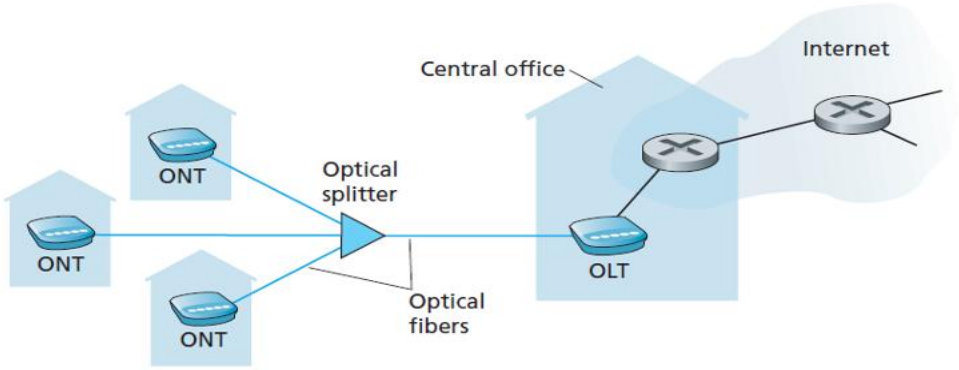


Fig: FTTH Internet access

- Figure shows FTTH using the PON distribution architecture. Each home has an optical network terminator (ONT), which is connected by dedicated optical fiber to a neighborhood splitter.
- The splitter combines a number of homes (typically less than 100) onto a single, shared optical fiber, which connects to an optical line terminator (OLT) in the telco's CO.
- The OLT, providing conversion between optical and electrical signals, connects to the Internet via a telco router.
- In the home, users connect a home router (typically a wireless router) to the ONT and access the Internet via this home router.
- In the PON architecture, all packets sent from OLT to the splitter are replicated at the splitter (similar to a cable head end).
- FTTH can potentially provide Internet access rates in the gigabits per second range.

iv) Dial-Up, and Satellite: *Dial-up* access over traditional phone lines is based on the same model as DSL—a home modem connects over a phone line to a modem in the ISP. Compared with DSL and other broadband access networks, dial-up access is excruciatingly slow at 56 kbps.

Satellite link can be used to connect a residence to the Internet at speeds of more than 1 Mbps; StarBand and HughesNet are two such satellite access providers.

v) Access in the Enterprise (and the Home): Ethernet and WiFi

Ethernet: On corporate and university campuses a local area network (LAN) is used to connect an end system to the edge router.

- There are many types of LAN technologies; Ethernet is the most prevalent access technology in corporate, university, and home networks.
- As shown in Figure, Ethernet users use twisted-pair copper wire to connect to an Ethernet switch.
- With Ethernet access, users typically have 100 Mbps access to the Ethernet switch, whereas servers may have 1 Gbps or even 10 Gbps access.

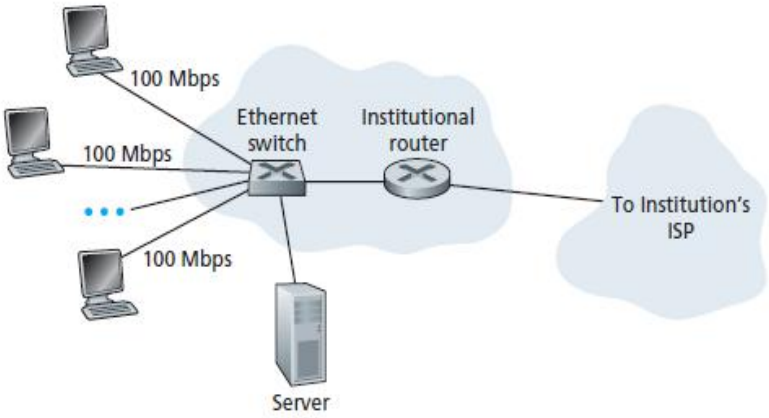


Fig: Ethernet Internet access

WiFi: Increasingly, however, people are accessing the Internet wirelessly from laptops, smart phones, tablets, and other devices.

- In a wireless LAN setting, wireless users transmit/receive packets to/from an access point that is connected into the enterprise's network (most likely including wired Ethernet), which in turn is connected to the wired Internet.
- A wireless LAN user must typically be within a few tens of meters of the access point.
- Wireless LAN access based on IEEE 802.11 technology, more colloquially known as *WiFi*, is now just about everywhere—universities, business offices, cafes, airports, homes, and even in airplanes.
- 802.11 today provides a shared transmission rate of up to 54 Mbps.

3. The Network Core

The network core—the mesh of packet switches and links that interconnects the Internet's end systems. Figure highlights the network core with thick, shaded lines.

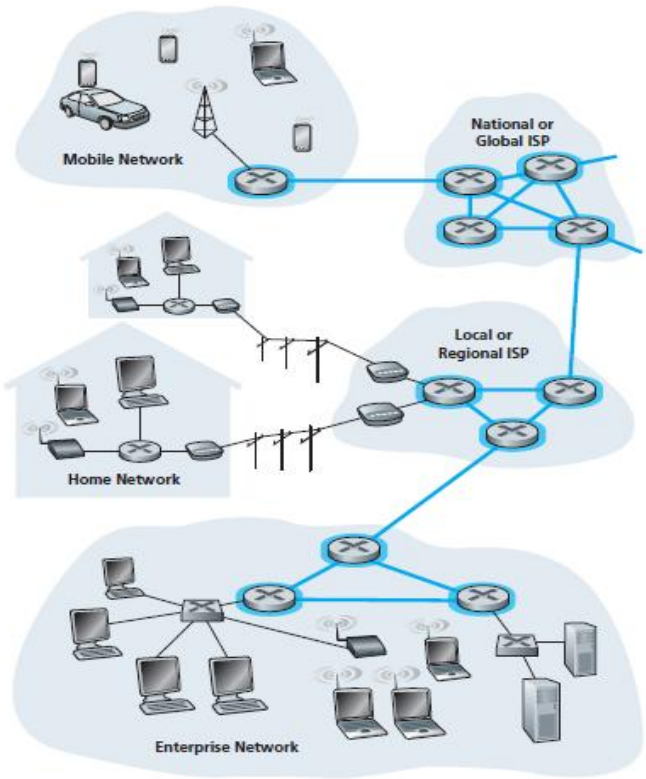


Figure: The network core

- The network edge refers to endpoints. The network core refers to the components that provide services to those at the edge.
- Routing and forwarding are the two key network core functions.

There are two fundamental approaches to moving data through a network of links and switches: **circuit switching** and **packet switching**.

Packet Switching

- In a network application, end systems exchange **messages** with each other. Messages can contain anything the application designer wants.
- Messages may perform a control function or can contain data, such as an email message, a JPEG image, or an MP3audio file.
- To send a message from a source end system to a destination end system, the source breaks long messages into smaller chunks of data known as **packets**.
- Between source and destination, each packet travels through communication links and **packet switches** (for which there are two types, **routers** and **linklayer switches**).
- Packets are transmitted over each communication link at a rate equal to the *full* transmission rate of the link.
- If a source end system or a packet switch is sending a packet of *L* bits over a link with transmission rate *R* bits/sec, then the time to transmit the packet is **L/R seconds**.

Store-and-Forward Transmission: Most packet switches use **store-and-forward transmission** at the inputs to the links.

- Store-and-forward transmission means that the packet switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link.
- To explore store-and-forward transmission in more detail, consider a simple network consisting of two end systems connected by a single router, as shown in Figure.

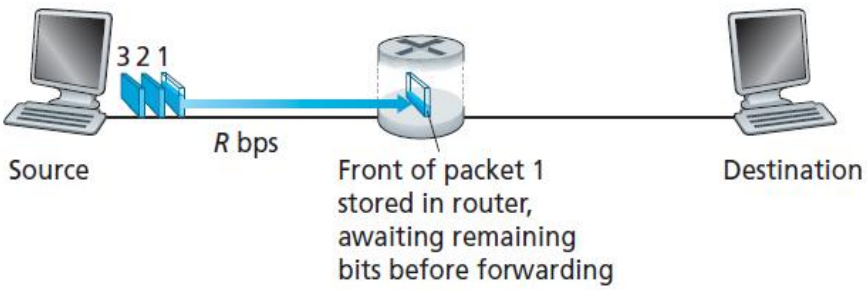


Fig: Store-and-forward packet switching

- A router will typically have many incident links, since its job is to switch an incoming packet onto an outgoing link.
- In this example, the source has three packets, each consisting of *L* bits, to send to the destination.
- The source has transmitted some of packet 1, and the front of packet 1has already arrived at the router. Because the router employs store-and-forwarding, at this instant of time, the router cannot transmit the bits it has received; instead it must first buffer (i.e., "store") the packet's bits.
- Only after the router has received *all* of the packet's bits can it begin to transmit (i.e., "forward") the packet onto the outbound link.

Let's now consider the general case of sending one packet from source to destination over a path consisting of *N* links each of rate *R*. Applying the same logic as above, we see that the end-to-end delay is:

$$d_{\text{end-to-end}} = N\frac{L}{R}$$

 Queuing Delays and Packet Loss: Each packet switch has multiple links attached to it. For each attached link, the packet switch has an **output buffer** (also called an **output queue**), which stores packets that the router is about to send into that link.

- The output buffers play a key role in packet switching. If an arriving packet needs to be transmitted onto a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer.
- Thus, in addition to the store-and-forward delays, packets suffer output buffer **queuing delays**. These delays are variable and depend on the level of congestion in the network.
- The amount of buffer space is finite, an arriving packet may find that the buffer is completely full with other packets waiting for transmission. In this case, **packet loss** will occur—either the arriving packet or one of the already-queued packets will be dropped.

Figure illustrates a simple packet-switched network.

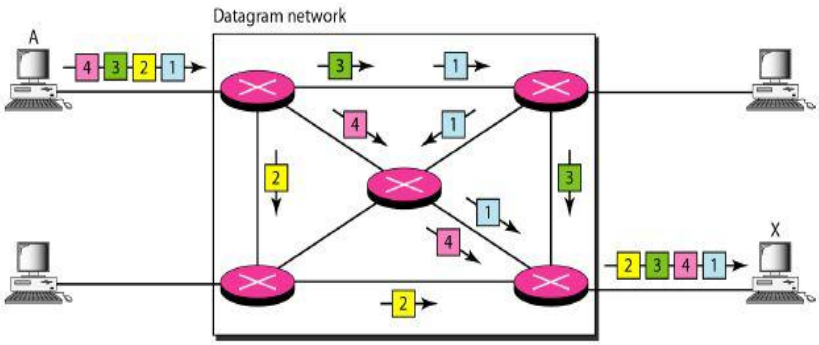


Fig: Packet switching

 Forwarding Tables and Routing Protocols: In the Internet, every end system has an address called an IP address. When a source end system wants to send a packet to a destination end system, the source includes the destination's IP address in the packet's header.

- When a packet arrives at a router in the network, the router examines a portion of the packet's destination address and forwards the packet to an adjacent router.
- More specifically, each router has a **forwarding table** that maps destination addresses (or portions of the destination addresses) to that router's outbound links.
- When a packet arrives at a router, the router examines the address and searches its forwarding table, using this destination address, to find the appropriate outbound link. The router then directs the packet to this outbound link.
- The Internet has a number of special **routing protocols** that are used to automatically set the forwarding tables. A routing protocol may, for example, determine the shortest path from each router to each destination and use the shortest path results to configure the forwarding tables in the routers.

Circuit Switching

- In circuit-switched networks, the resources needed along a path (buffers, link transmission rate) to provide for communication between the end systems are *reserved* for the duration of the communication session between the end systems.
- In packet-switched networks, these resources are *not* reserved; a session's messages use the resources on demand, and as a consequence, may have to wait (that is, queue) for access to a communication link.
- Traditional telephone networks are examples of circuit-switched networks.
- A circuit-switched network is made of a set of switches connected by physical

links, in which each link is divided into n channels.

- Each link is divided into n (n is 3 in the figure) channels by using FDM or TDM.

 Three phases: A circuit-switched network consists of ***3 phases***: 1) Setup phase (establish),2) Data transfer phase (transfer), 3) Tear down phase (disconnect).

- **Setup Phase**: Before the two parties can communicate, a dedicated circuit needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches.
- **Data-Transfer Phase:** After the establishment of the dedicated circuit (channels), the two parties can transfer data.
- **Teardown Phase:** When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

Figure illustrates a circuit-switched network. In this network, the four circuit switches are interconnected by four links. Each of these links has four circuits, so that each link can support four simultaneous connections.

- The hosts are each directly connected to one of the switches. When two hosts want to communicate, the network establishes a dedicated **end-to-end connection** between the two hosts.

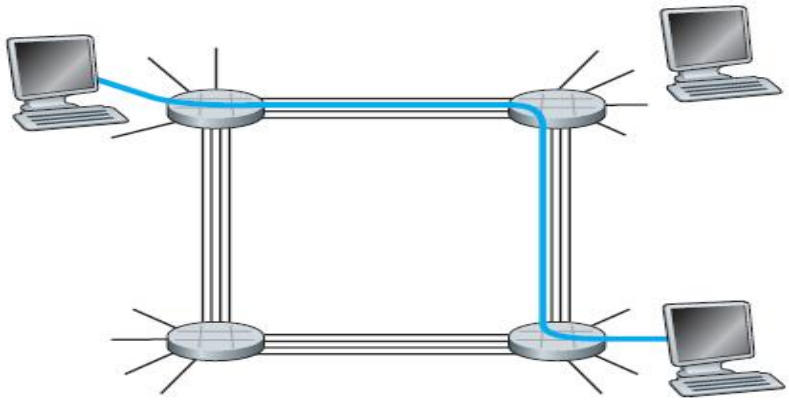


Fig: A simple circuit-switched network consisting of four switches and four links

For example: When end system A needs to communicate with end system B, system A needs to request a connection to B that must be accepted by all switches as well as by B itself. This is called the ***setup phase***; after the dedicated path made of connected circuits (channels) is established, the ***data-transfer*** phase can take place. After all data have been transferred, the circuits are **tearing down.**
In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer until the teardown phase.

 Multiplexing in Circuit-Switched Networks: A circuit in a link is implemented with either **frequency-division multiplexing (FDM)** or **time-division multiplexing (TDM)**.

With ***FDM***, the frequency spectrum of a link is divided up among the connections established across the link.

- The link dedicates a frequency band to each connection for the duration of the connection. The width of the band is called, not surprisingly, the **bandwidth**.
- FM radio stations also use FDM to share the frequency spectrum between 88 MHz and 108 MHz, with each station being allocated a specific frequency band.

For a ***TDM*** link, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a

connection across a link, the network dedicates one time slot in every frame to this connection.

Figure shows FDM and TDM for a specific network link supporting up to four circuits. For FDM, the frequency domain is segmented into four bands, each of bandwidth 4 kHz. For TDM, the time domain is segmented into frames, with four time slots in each frame

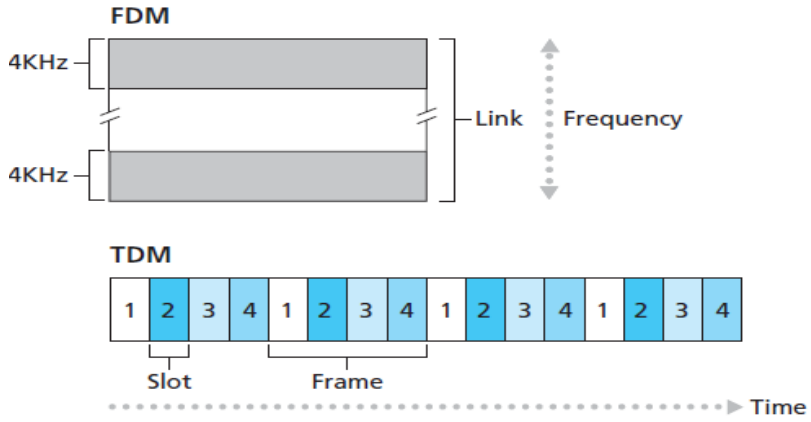


Fig: FDM & TDM

4. Delay, Loss, and Throughput in Packet-Switched Networks

Overview of Delay in Packet-Switched Networks

A packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node (host or router) to the subsequent node (host or router) along this path, the packet suffers from several types of delays at *each* node along the path.

The most important of these delays are the **nodal processing delay, queuing delay, transmission delay,** and **propagation delay**; together, these delays accumulate to give a **total nodal delay**.

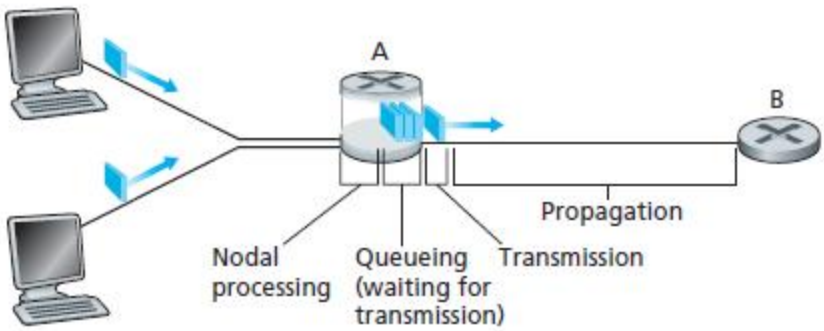


Fig: The nodal delay at router A

Types of Delay: The end-to-end route between source and destination, a packet is sent from the upstream node through router A to router B. Our goal is to characterize the nodal delay at router A.

i) Processing Delay: The time required to examine the packet's header and determine where to direct the packet is part of the **processing delay**.

- The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream node to router A.
- Processing delays in high-speed routers are typically on the order of microseconds or less. After this nodal processing, the router directs the packet to the queue that precedes the link to router B.

ii) **Queuing Delay:** At the queue, the packet experiences a **queuing delay** as it waits to be transmitted onto the link.

- The length of the queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission onto the link.
- If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay will be zero.

On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long.

- Queuing delays can be on the order of microseconds to milliseconds in practice.

iii) **Transmission Delay:** This is the amount of time required to push (that is, transmit) all of the packet's bits into the link.

- Denote the length of the packet by *L* bits, and denote the transmission rate of the link from router A to router B by *R* bits/sec.
- For example, for a 10 Mbps Ethernet link, the rate is *R* = 10 Mbps; for a 100 Mbps Ethernet link, the rate is *R* = 100 Mbps.
- The **transmission delay** is *L/R.* Transmission delays are typically on the order of microseconds to milliseconds in practice.

iv) **Propagation Delay:** Once a bit is pushed into the link, it needs to propagate to router B. The time required to propagate from the beginning of the link to router B is the **propagation delay**.

- The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link (that is, fiber optics, twisted-pair copper wire, and so on)
- The propagation delay is the distance between two routers divided by the propagation speed. That is, the propagation delay is *d/s,* where *d* is the distance between router A and router B and *s* is the propagation speed of the link. Propagation delays are on the order of milliseconds.

The total ***nodal delay*** is given by

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Queuing Delay and Packet Loss

The queuing delay can vary from packet to packet. For example, if 10 packets arrive at an empty queue at the same time, the first packet transmitted will suffer no queuing delay, while the last packet transmitted will suffer a relatively large queuing delay (while it waits for the other nine packets to be transmitted).

On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long.

The ratio *of* the **traffic intensity is** ***La/R***

Let ***a*** denote the average rate at which packets arrive to the queue (*a* is units of packets/sec), ***R*** is the transmission rate, i.e., it is the transmission rate (in bits/sec) and **L** is average packet length (in bits).

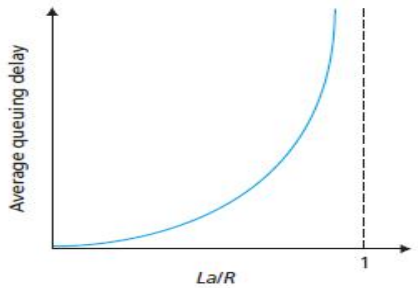


Fig: Dependence of average queuing delay on traffic intensity

The fact that as the traffic intensity approaches 1, the average queuing delay increases rapidly. A small percentage increase in the intensity will result in a much larger percentage-wise increase in delay.

Packet Loss: we have assumed that the queue is capable of holding an infinite number of packets. In reality a queue preceding a link has finite capacity, although the queuing capacity greatly depends on the router design and cost. Because the queue capacity is finite, packet delays do not really approach infinity as the traffic intensity approaches 1. Instead, a packet can arrive to find a full queue.

With no place to store such a packet, a router will **drop** that packet; that is, the packet will be **lost**.

- **End-to-End Delay:** nodal delay is the delay at a single router. Let's now consider the total delay from source to destination. To get a handle on this concept, suppose there are *N*-1 routers between the source host and the destination host.

The nodal delays accumulate and give an end-to-end delay,

$$d_{\text{end-end}} = N\,(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

Throughput in Computer Networks

In data transmission, network throughput is **the amount of data moved successfully from one place to another in a given time period**, and typically measured in bits per second (bps), as in megabits per second (Mbps) or gigabits per second (Gbps).

Network throughput **refers to how much data can be transferred from source to destination within a given timeframe.**

To define throughput, consider transferring a large file from Host A to Host B across a computer network.

The **instantaneous throughput** at any instant of time is the rate (in bits/sec) at which Host B is receiving the file.

If the file consists of *F* bits and the transfer takes *T* seconds for Host B to receive all *F* bits, then the **average throughput** of the file transfer is *F/T* bits/sec.

Example: Figure (a) shows two end systems, a server and a client, connected by two communication links and a router. Consider the throughput for a file transfer from the server to the client.

Let *Rs* denote the rate of the link between the server and the router; and *Rc* denote the rate of the link between the router and the client.

For this simple two-link network, the throughput is min{*Rc*, *Rs*}, that is, it is the transmission rate of the **bottleneck link**. Having determined the throughput, we can now approximate the time it takes to transfer a large file of *F* bits from server to client as *F*/min{*Rs*, *Rc*}.

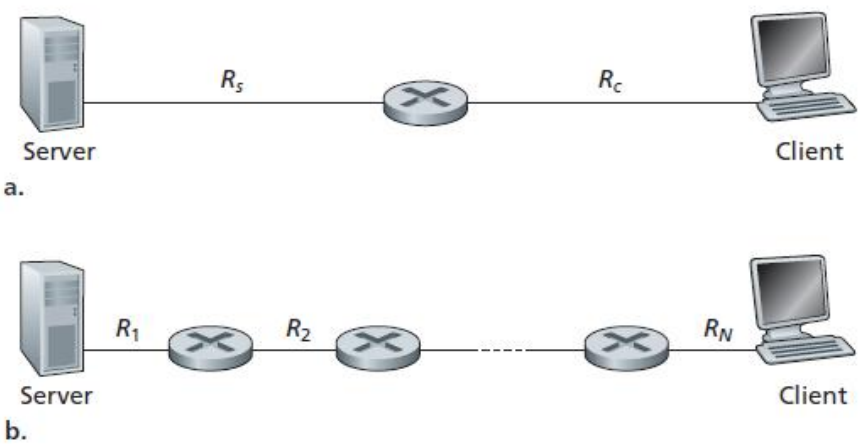


Fig: Throughput for a file transfer from server to client

Figure (b) now shows a network with *N* links between the server and the client, with the transmission rates of the *N* links being *R*1, *R*2,..., *RN*. Applying the same analysis as for the two-link network, we find that the throughput for a file transfer from server to client is min{*R*1, *R*2,..., *RN*},

5. Reference Models

Layered Architecture or Protocol Layering

A layered architecture allows us to discuss a well-defined, specific part of a large and complex system. In layered architecture of network model, one whole network process is divided into small tasks. Each small task is then assigned to a particular layer which works dedicatedly to process the task only. Every layer does only specific work.
The layer provides the same service to the layer above it, and uses the same services from the below it.

In data communication and networking, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to be able to communicate effectively. When communication is simple, we may need only one simple protocol; when the communication is complex, we may need to divide the task between different layers, in which case we need a protocol at each layer, or protocol layering.

Scenarios

Let us develop two simple scenarios to better understand the need for protocol layering.
First Scenario: In the first scenario, communication is so simple that it can occur in only one layer. Assume Maria and Ann are neighbors with a lot of common ideas. Communication between Maria and Ann takes place in one layer.

Fig: A single-layer protocol

Second Scenario: In the second scenario, we assume that Ann is offered a higher-level position in her company, but needs to move to another branch located in a city very far from Maria. The two friends still want to continue their communication and exchange ideas by using protocol layering.

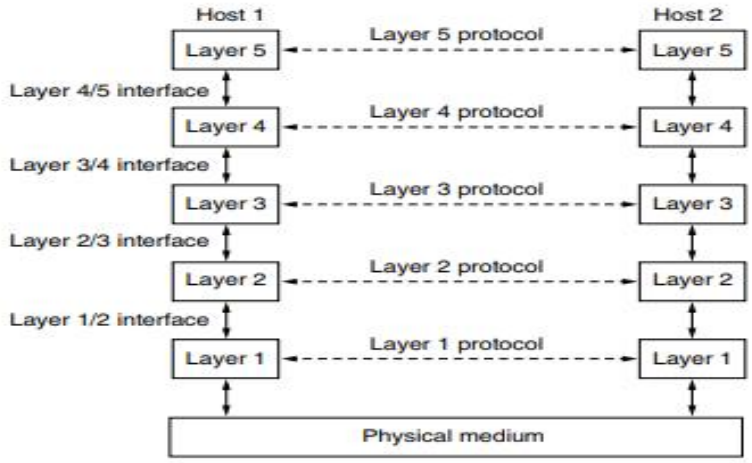


Fig: multiple protocols layering

Two models have been devised to define computer network operations: the ***TCP/IP protocol suite*** and the ***OSI model.*** The protocol layering is used in both models.

Reference Models

i) OSI model

The OSI model is based on a proposal developed by the International Standards Organization (ISO). The model is called the ISO OSI (Open Systems Interconnection), which allows different systems to communicate.

An open system is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture. The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software.

The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems.

It consists of ***seven layers***: 1. Physical Layer, 2. Data link Layer, 3. Network Layer, 4.transport Layer, 5. Session Layer, 6. Presentation layer, 7. Application Layer.

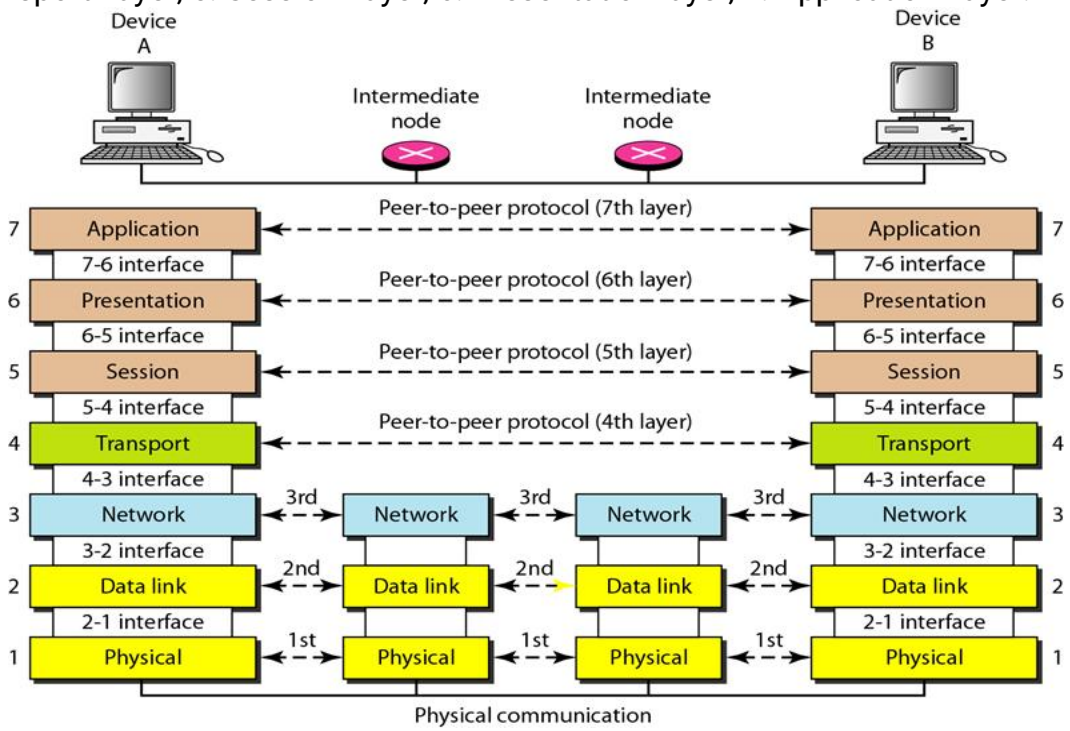


Fig: The interaction between layers in the OSI model

i) **Physical layer:** the physical layer is responsible for movement of individual bits from one node to the next.

- The physical layer required to carry a bit stream over a physical medium.
- It deals with the mechanical and electrical specifications of the interface and transmission medium.

Responsibilities of physical layer:

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

Physical topology: The physical topology defines how devices are connected to make a network.

Transmission mode: The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex.

ii) **Data Link Layer:** The data link layer is responsible for moving frames from one node to the next.

Frame: Frame is a series of bits that form a unit of data.

Responsibilities of the data link layer:

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

Physical addressing: The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN. It is included in the frame used by the data link layer. It is the lowest-level address.

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 adding a mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplicate frames.

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.

iii) **Network Layer:** The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links).

The network layer is responsible for the delivery of individual packets from the source host to the destination host.

Responsibilities of the network layer

Logical addressing: Addressing system to help to differentiate the 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 internetworks (network of networks) or a large network, the connecting devices (called routers or switches) route or switch the packets to their final destination.

iv) **Transport Layer**: The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on a host.

Responsibilities of the transport layer:

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 includes a type of address called a service-point address (or port

address).
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.
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.

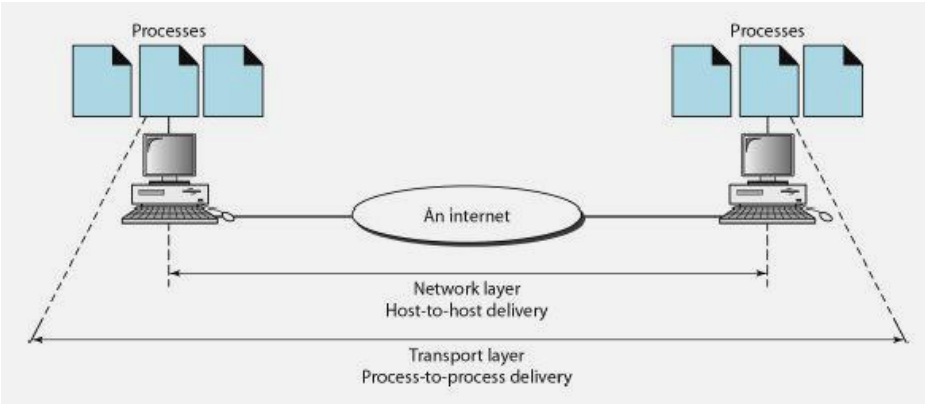


Fig: process-to-process delivery of a message

v) Session Layer: The session layer is responsible for dialog control and synchronization. The session layer is the network dialog controller. It establishes, maintains, and synchronizes the interaction among communicating systems
Responsibilities of the session layer:
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.
For example, if a system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be resent.

vi) Presentation Layer: The presentation layer is responsible for translation, compression, and encryption. The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.
Responsibilities of the presentation layer:
Translation: The presentation layer at the sender machine 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: To carry sensitive information, a system must be able to ensure privacy. 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.
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.

vii) Application Layer: The application layer is responsible for providing services to

the user.
The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services.

Services provided by the application layer:

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

Mail services: This application provides the basis for e-mail forwarding and storage.

Directory services: This application provides distributed database sources and access for global information about various objects and services.

ii) TCP/IP Protocol Suite

The TCPIIP protocol suite was developed prior to the OSI model. The original TCP/IP protocol suite was defined as having ***four layers***: Host-To-Network (Network Interface), Internet, Transport, and Application.

The 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, and the application layer is roughly doing the job of the session, presentation, and application layers.

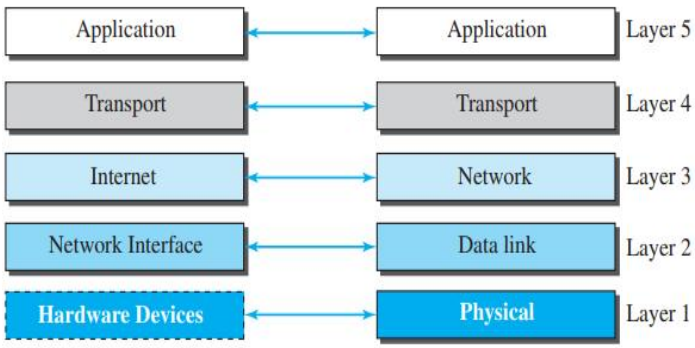


Fig: Layers in the TCP/IP protocol suite

- **Physical Layer:** We can say that the physical layer is responsible for carrying individual bits in a frame across the link. The physical layer is the lowest level in the TCP/IP protocol suite. There is a hidden layer, the transmission media, under the physical layer. Two devices are connected by a transmission medium (cable or air).The transmission medium does not carry bits; it carries electrical or optical signals.

- **Data-link Layer:** the data-link layer is responsible for taking the datagram and moving it across the link. The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN. We can also have different protocols used with any link type.

TCP/IP does not define any specific protocol for the data-link layer, but it uses the HDLC and PPP protocols. It supports all the standard and proprietary protocols.

- **Network Layer:** The network layer is responsible for creating a connection between the source computer and the destination computer. The communication at the network layer is host-to-host. There can be several routers from the source to the destination; the routers in the path are responsible for choosing the best route for each packet.

In network layer the ***main protocol*** is **Internet Protocol** (IP), which defines the format of the packet, called a datagram at the network layer. IP also defines the format and the structure of addresses used in this layer.
The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks. The **Internet Control Message Protocol** (ICMP) helps IP to report some problems when routing a packet. The **Internet Group Management Protocol** (IGMP) is another protocol that helps IP in multitasking. The **Dynamic Host Configuration Protocol** (DHCP) helps IP to get the network-layer address for a host. The **Address Resolution Protocol** (ARP) is a protocol that helps IP to find the link-layer address of a host or a router when its network-layer address is given.

- **Transport Layer:** The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transport layer packet called a segment or a user datagram.
The main protocol, **Transmission Control Protocol** (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data. It creates a logical pipe between two TCPs for transferring a stream of bytes.
The other common protocol, **User Datagram Protocol** (UDP), is a connectionless protocol that transmits user datagram without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one.

- **Application Layer:** The logical connection between the two application layers is end to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers.
The **Hypertext Transfer Protocol** (HTTP) is a vehicle for accessing the World Wide Web (WWW). The **Simple Mail Transfer Protocol** (SMTP) is the main protocol used in electronic mail (e-mail) service. The **File Transfer Protocol** (FTP) is used for transferring files from one host to another. The **Terminal Network** (TELNET) and **Secure Shell** (SSH) are used for accessing a site remotely. The **Simple Network Management Protocol** (SNMP) is used by an administrator to manage the Internet at global and local levels. The **Domain Name System** (DNS) is used by other protocols to find the network-layer address of a computer.

6. Transmission Media

Introduction: Transmission media are actually located below the physical layer and are directly controlled by the physical layer. We could say that transmission media belong to layer zero. Figure shows the position of transmission media in relation to the physical layer.

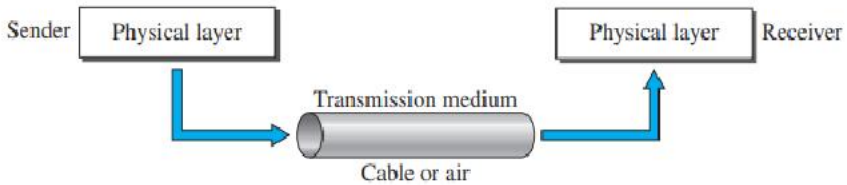


Fig: Transmission medium and physical layer

A transmission medium can be broadly defined as anything that can carry information from a source to a destination. It is also called physical medium.
The transmission medium is usually free space, metallic cable, or fiber-optic cable. The information is usually a signal that is the result of a conversion of data from another form.

Transmission media can be divided into **two broad categories**: guided and unguided.

Guided media include twisted-pair cable, coaxial cable, and fiber-optic cable. Unguided medium is free space.

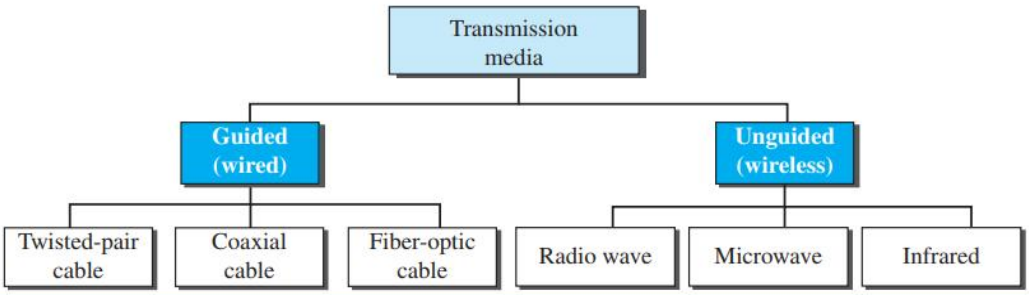


Fig: Classes of transmission media

i) Guided Media: Wired

Guided media, which are those that provide a channel from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these 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, as shown in Figure

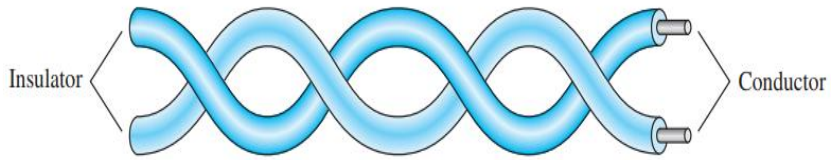


Fig: Twisted-pair cable

One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals.

Unshielded Versus Shielded Twisted-Pair Cable: The most common twisted-pair cable used in communications is referred to as unshielded twisted-pair (UTP).

IBM has also produced a version of twisted-pair cable for its use, called shielded twisted-pair (STP). STP cable has a metal foil or braided mesh covering that encases each pair of insulated conductors. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive.

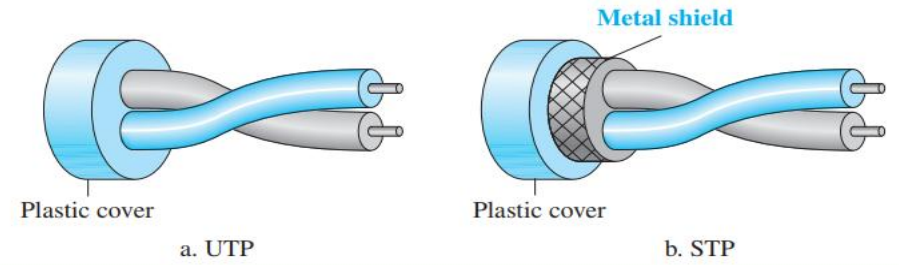


Fig: UTP and STP cables

Performance: One way to measure the performance of twisted-pair cable is to compare attenuation versus frequency and distance. A twisted-pair cable can pass a wide range of frequencies.

Applications: Twisted-pair cables are used in telephone lines to provide voice and data channels.

- Local-area networks also use twisted-pair cables.

 Coaxial Cable: Coaxial cable (or coax) carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two.

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.

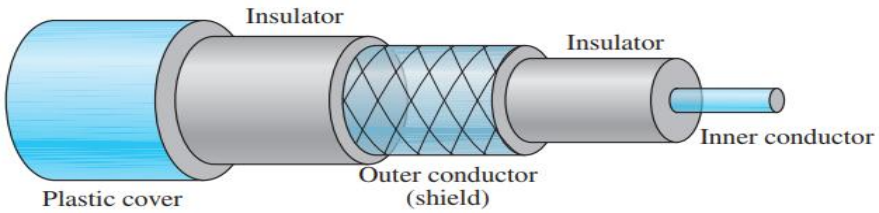


Fig: Coaxial cable

Performance we measure the performance of a coaxial cable. The attenuation is much higher in coaxial cable than in twisted-pair cable. In other words, 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 where a single coaxial network could carry 10,000 voice signals.

- Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps.
- However, coaxial cable in telephone networks has largely been replaced today with fiber optic cable.
- Cable TV networks also use coaxial cables. Later, however, cable TV providers replaced most of the media with fiber-optic cable.
- Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs.

 Fiber-Optic Cable: A fiber-optic cable is made of glass or plastic and transmits signals in the form of light.

Light travels in a straight line as long as it is moving through a single uniform substance. If a ray of light traveling through one substance suddenly enters another substance (density), the ray changes direction.

- Figure shows how a ray of light changes direction when going from a more dense to a less dense substance. As the figure shows, if the angle of incidence I is less than the critical angle, the ray refracts and moves closer to the surface.
- If the angle of incidence is equal to the critical angle, the light bends along the interface.
- If the angle is greater than the critical angle, the ray reflects (makes a turn) and travels again in the denser substance.

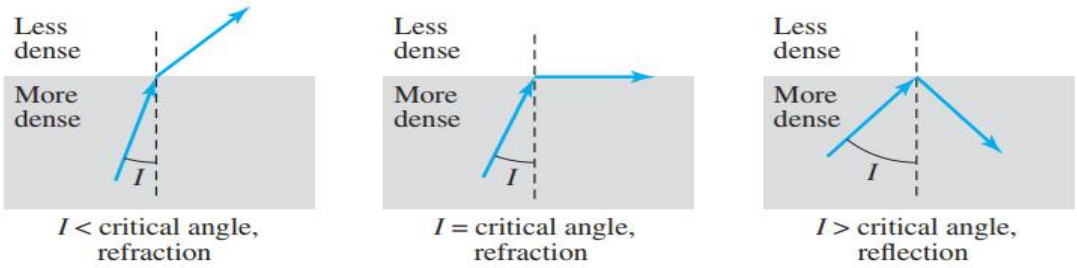


Fig: Bending of light ray

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.

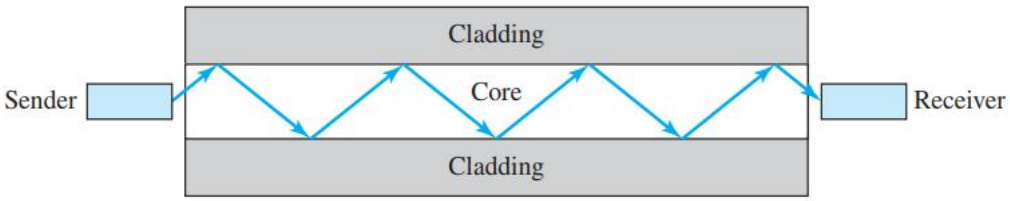


Fig: Optical fiber

Cable Composition: Figure shows the composition of a typical fiber-optic cable. The outer jacket is made of 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.

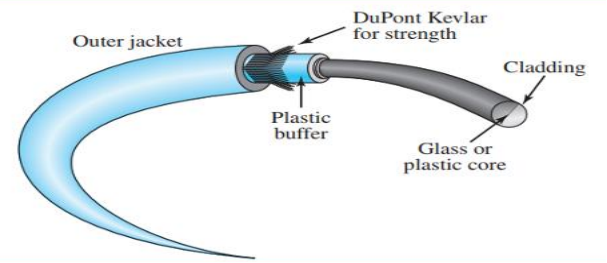


Fig: Fiber construction

Performance: Attenuation is flatter than in the case of twisted-pair cable and coaxial cable. The performance is such that we need fewer repeaters when we use fiber-optic cable.

Advantages and Disadvantages of Optical Fiber:
Advantages: Fiber-optic cable has several advantages over metallic cable (twisted-pair or coaxial).

- Higher bandwidth.
- Less signal attenuation..
- Immunity to electromagnetic interference.
- Resistance to corrosive materials.
- Light weight.

.
Disadvantages There are some disadvantages in the use of optical fiber.

- Installation and maintenance.
- Unidirectional light propagation.
- Cost.

ii) Unguided Media: Wireless

Unguided medium transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication. Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them.
Figure shows the part of the electromagnetic spectrum, ranging from 3 kHz to 900 THz, used for wireless communication.

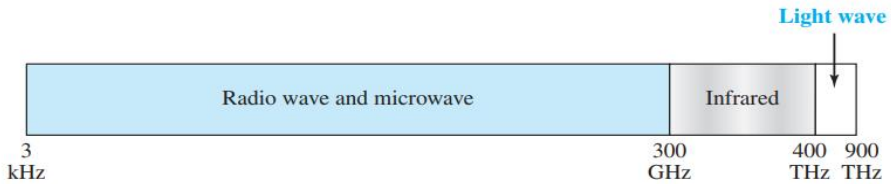


Fig: Electromagnetic spectrum for wireless communication

Unguided signals can travel from the source to the destination in several ways: ground propagation, sky propagation, and line-of-sight propagation, as shown in Figure 7.18.

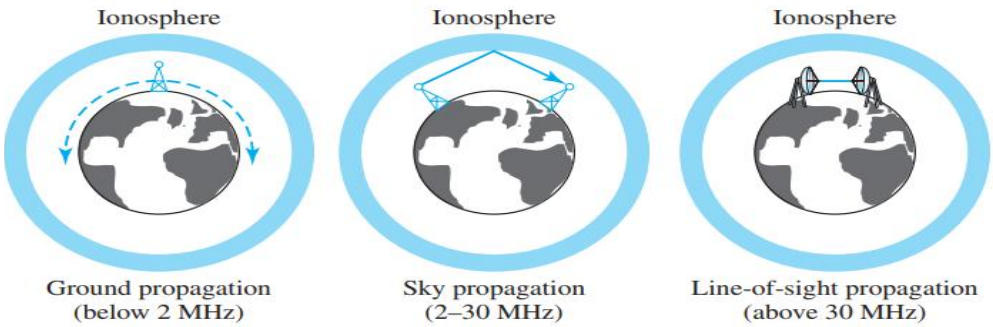


Fig: Propagation methods

In ***ground propagation***, radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet.
In ***sky propagation***, higher-frequency radio waves radiate upward into the ionosphere (the layer of atmosphere where particles exist as ions) where they are reflected back to earth. This type of transmission allows for greater distances with lower output power.
In ***line-of-sight propagation***, very high-frequency signals are transmitted in straight lines directly from antenna to antenna.

 Radio Waves: The electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves.

- Radio waves are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. A sending antenna sends waves that can be received by any receiving antenna.
- Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio.
- Radio waves, particularly those of low and medium frequencies, can penetrate walls. It is an advantage because, for example, an AM radio can receive signals inside a building.

Omnidirectional Antenna Radio waves use omnidirectional antennas that send out signals in all directions. Figure shows an omnidirectional antenna.

Fig: Omnidirectional antenna

Applications The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers.
AM and FM radio, television, cordlessphones, and paging are examples of multicasting.

 Microwaves Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. This means that the sending and receiving antennas need to be aligned.
The following describes some ***characteristics*** of microwave propagation:

- Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall.
- Very high-frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.

Unidirectional Antenna 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.

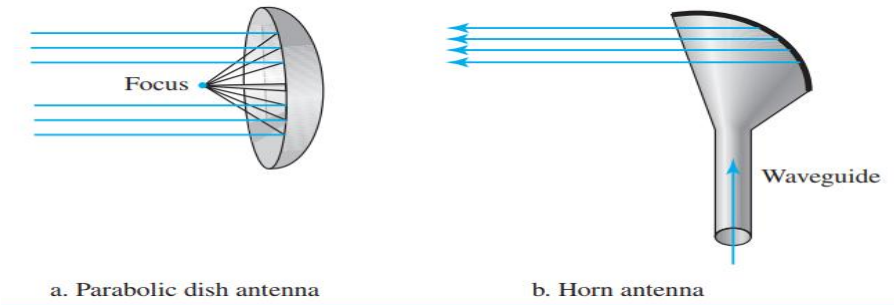


Fig: Unidirectional antennas

Applications: Microwaves, due to their unidirectional properties, are very useful when unicast (oneto-one) communication is needed between the sender and the receiver.
They are used in cellular phones, satellite networks, and wireless LANs.
 Infrared: Infrared waves, with frequencies from 300 GHz to 400 THz. It can be used for short-range communication. 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. When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors.
Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation

7. Example Networks

i) Internet

A Brief History: A network is a group of connected communicating devices such as computers and printers. The Internet collaboration of more than hundreds of thousands of interconnected networks.
Private individuals as well as various organizations such as government agencies, schools, research facilities, corporations, and libraries in more than 100 countries use the Internet. Millions of people are users. The extraordinary communication system only came into being in 1969.

In the *mid-1960s*, Computers from different manufacturers were unable to communicate with one another. The Advanced Research Projects Agency (ARPA) in the Department of Defense (DoD) was interested in finding a way to connect computers.

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for ARPANET, a small network of connected computers. The idea was that each host computer would be attached to a specialized computer, called an interface message processor (IMP).

By *1969*, ARPANET was a reality. Four nodes, at the Universities were connected via the IMPs to form a network. Software called the Network Control Protocol (NCP) provided communication between the hosts.

In *1972*, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the Internetting Project.

Cerf and Kahn's landmark *1973* paper outlined the protocols to achieve end-to-end delivery of packets. This paper on Transmission Control Protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway.
After that split TCP into two protocols: Transmission Control Protocol (TCP) and Internetworking Protocol (IP).

The Internet Today: Today most end users who want Internet connection use the services of Internet service providers (ISPs). There are international service providers, national service providers, regional service providers, and local service providers.

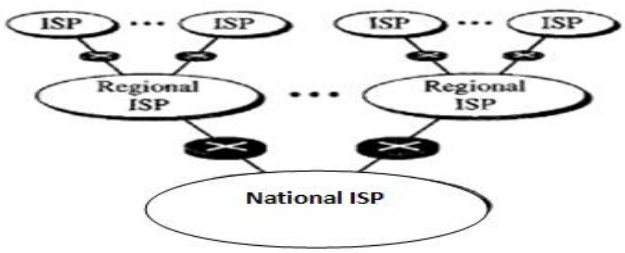


Fig: Structure of a national ISP

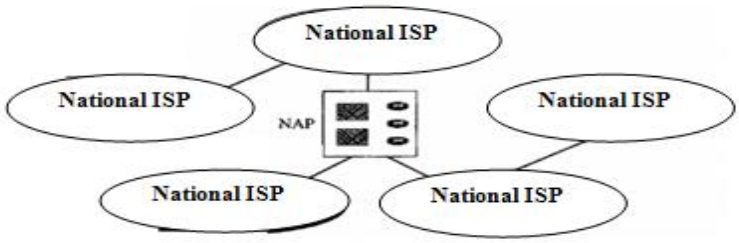


Fig: Interconnection of national ISPs

ii) Third-Generation Mobile Phone Networks

People love to talk on the phone even more than they like to surf the Internet, and this has made the mobile phone network the most successful network in the world. It has more than four billion subscribers worldwide.
First-generation mobile phone systems transmitted voice calls as continuously varying (analog) signals rather than sequences of (digital) bits. **AMPS** (**Advanced Mobile Phone System**), which was deployed in the United States in 1982, was a widely used first generation system.
Second-generation mobile phone systems switched to transmitting voice calls in digital form to increase capacity, improve security, and offer text messaging. **GSM** (**Global System for Mobile communications**), which was deployed starting in 1991 and has become the most widely used mobile phone system in the world, is a 2G system.
The *third generation*, or 3G, systems were initially deployed in 2001 and offer both digital voice and broadband digital data services. **UMTS** (**Universal Mobile Telecommunications System**), also called **WCDMA** (**Wideband Code Division Multiple Access**), is the main 3G system that is being rapidly deployed worldwide.

iii) Wireless LANs: 802.11

The wireless LAN standard was dubbed 802.11. A common slang name for it is **WiFi** but it is an important standard and deserves respect, so we will call it by its proper name, 802.11.

802.11 networks are made up of clients, such as laptops and mobile phones, and infrastructure called **APs** (**access points**) that is installed in buildings. Access points are sometimes called **base stations**. The access points connect to the wired network, and all communication between clients goes through an access point.

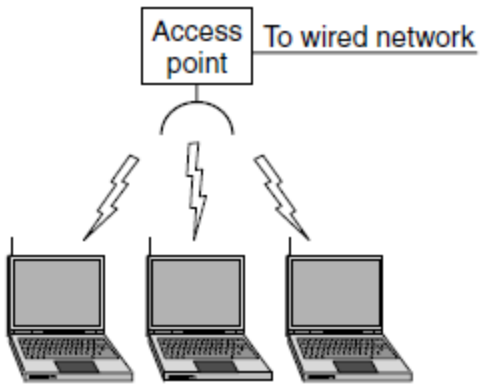


Fig: Wireless network with an access point

COMPUTER NETWORKS

UNIT - 2: DATA LINK LAYER

Syllabus:

Data link layer: Design issues, Framing: fixed size framing, variable size framing, flow control, error control, error detection and correction codes, CRC, Checksum: idea, one's complement internet checksum services provided to Network Layer,

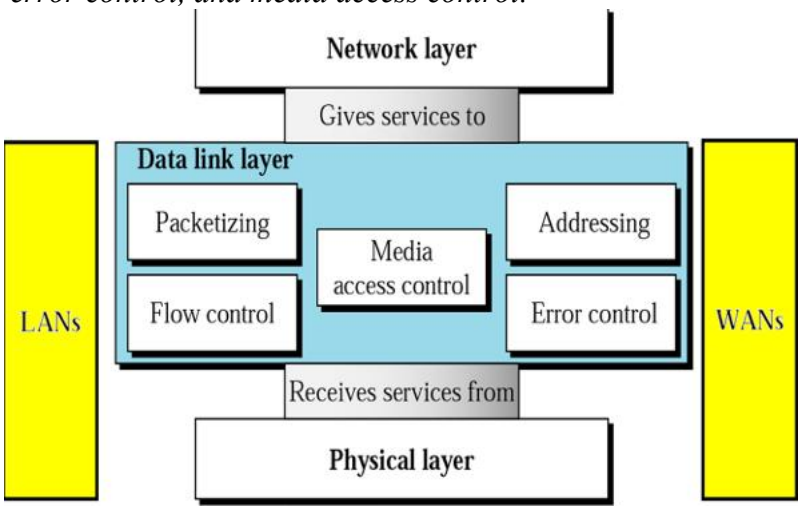
Elementary Data Link Layer protocols: simplex protocol, Simplex stop and wait, Simplex protocol for Noisy Channel.

Sliding window protocol: One bit, Go back N, Selective repeat-Stop and wait protocol, Data link layer in HDLC: configuration and transfer modes, frames, control field,

Point to point protocol (PPP): framing, transition phase, multiplexing, multi-link PPP.

DATA LINK LAYER

The data link layer transforms the physical layer, a raw transmission facility, to a link responsible for node-to-node (hop-to-hop) communication. Specific responsibilities of the data link layer include *framing, addressing, flow control, error control, and media access control.*

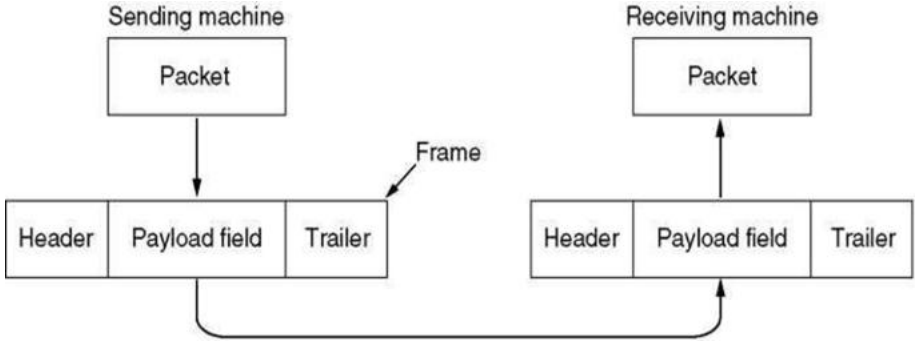


DATA LINK LAYER DESIGN ISSUES

The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:

1. Providing a well-defined service interface to the network layer.
2. Dealing with transmission errors.
3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig.



SERVICES PROVIDED TO THE NETWORK LAYER

The function of the data link layer is to provide services to the network layer. The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine.

The data link layer can be designed to offer various services. The actual services that are offered vary from protocol to protocol. Three reasonable possibilities that we will consider in turn are:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

1. Unacknowledged connectionless service:

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. Ethernet is a good example of a data link layer that provides this class of service. No logical connection is established beforehand or released afterward. If a frame is lost due to noise on the line attempt is made to detect the loss or recover from it in the data link layer. This class of service is appropriate when the error rate is very low, so recovery is left to higher layers.

2. Acknowledged connectionless service:

When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. In this way, the sender knows whether a frame has arrived correctly or been lost. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems. 802.11 (WiFi) is a good example of this class of service.

3. Acknowledged connection-oriented service:

The most sophisticated service the data link layer can provide to the network layer is connection-oriented service. With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

When connection-oriented service is used, transfers go through three distinct phases. In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not. In the second phase, one or more frames are actually transmitted. In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

FRAMING

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. If the channel is noisy, as it is for most wireless and some wired links, the physical layer will add some redundancy to its signals to reduce the bit error rate to a tolerable level. However, the bit stream received by the data link layer is not guaranteed to be error free. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the data link layer to detect and, if necessary, correct errors.

The data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted.

When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

DLL translates the physical layer's raw bit stream into discrete units (messages) called **frames.**

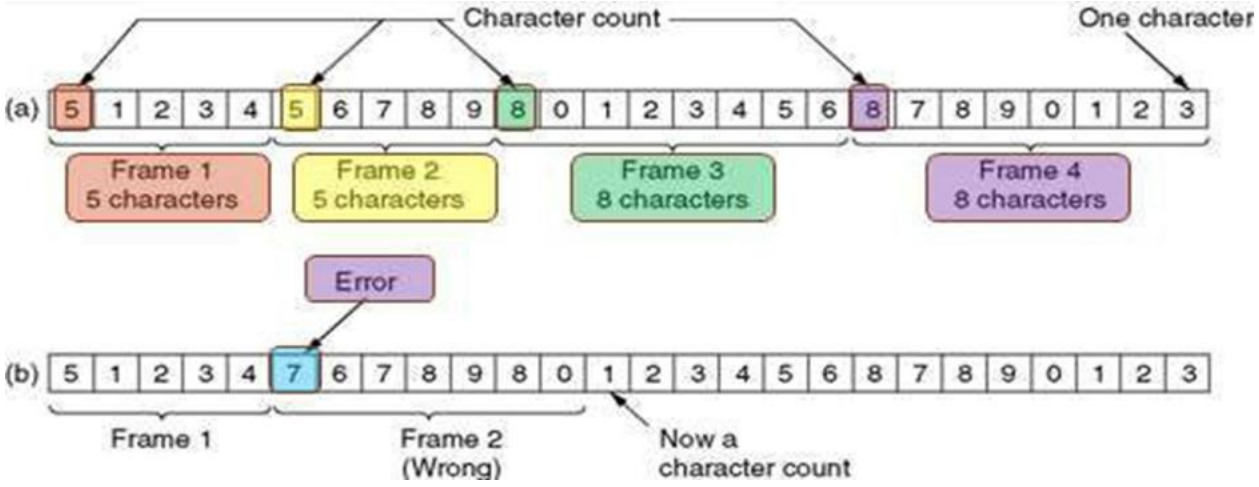
A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at **four methods:**

1. **Byte count.**
2. **Flag bytes with byte stuffing.**
3. **Flag bits with bit stuffing.**
4. **Physical layer coding violations.**

1. Byte count (Character Count) :

This framing method uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is. This technique is shown in Fig.(a) For four small example frames of sizes 5, 5, 8, and 8 bytes, respectively.

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the byte count of 5 in the second frame of Fig.(b) becomes a 7 due to a single bit flip, the destination will get out of synchronization. It will then be unable to locate the correct start of the next frame.

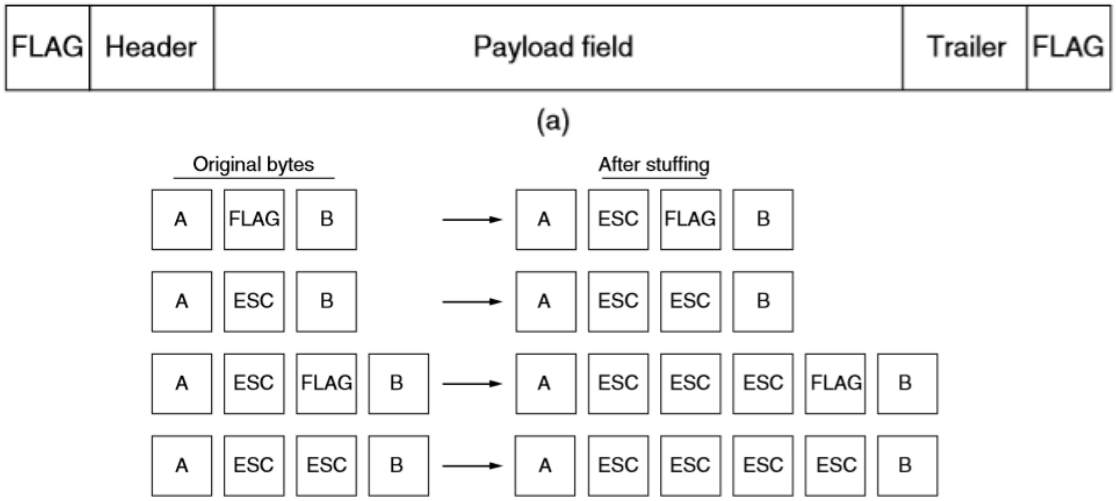


2. Flag bytes with byte stuffing:

This framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. Often the same byte, called a **flag byte**, is used as both the starting and ending delimiter. This byte is shown in Fig.(a) as **FLAG.** Two consecutive flag bytes indicate the end of one frame and the start of the next. Thus, if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame.

However, there is a still a problem we have to solve. It may happen that the flag byte occurs in the data, especially when binary data such as photographs or songs are being transmitted. This situation would interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each ''accidental'' flag byte in the data.

The data link layer on the receiving end removes the escape bytes before giving the data to the network layer. This technique is called **byte stuffing.**



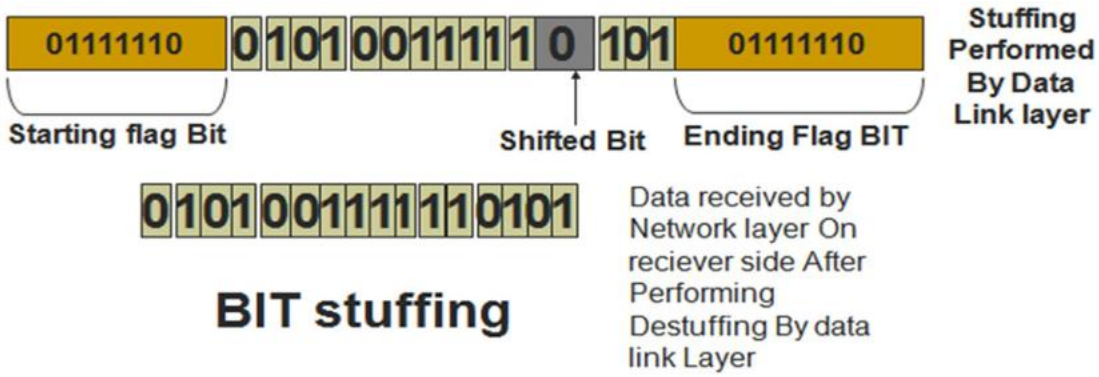
Four examples of byte sequences before and after byte stuffing.

3. Flag bits with bit stuffing:

Framing can be also be done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size. It was developed for the once very popular **HDLC (Highlevel Data Link Control)**

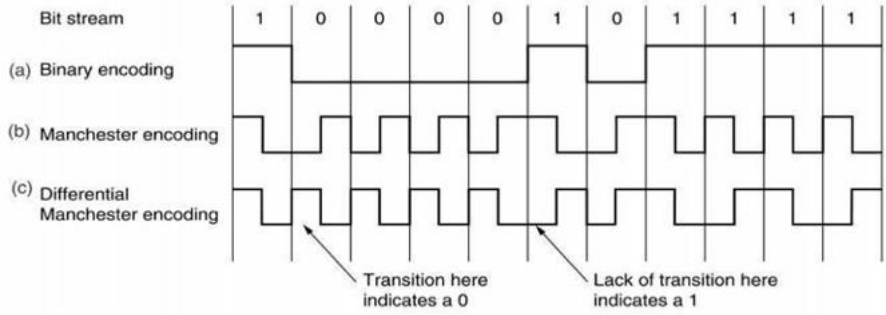
protocol. Each frame begins and ends with a special bit pattern, 01111110 or 0x7E in hexadecimal. This pattern is a flag byte. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This **bit stuffing** is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data. It also ensures a minimum density of transitions that help the physical layer maintain synchronization. USB (Universal Serial Bus) uses bit stuffing for this reason.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110. Figure gives an example of bit stuffing.



4. Physical layer coding violations:

- This Framing Method is used only in those networks in which Encoding on the Physical Medium contains some redundancy.
- Some LANs encode each bit of data by using two Physical Bits i.e. Manchester coding is used. Here, Bit 1 is encoded into high- low (10) pair and Bit 0 is encoded into low-high (01) pair.
- The scheme means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries. The combinations high-high and low-low are not used for data but are used for delimiting frames in some protocols.



ERROR CONTROL:

Error control is concerned with ensuring that all frames are eventually delivered (possibly in order) to a destination. How? Three items are required.

- **Acknowledgements:** Typically, reliable delivery is achieved using the "acknowledgments with retransmission" paradigm, whereby the receiver. returns a special acknowledgment (ACK) frame to the sender indicating the correct receipt of a frame. In some systems, the receiver also returns a negative acknowledgment (NACK) for incorrectly-received frames. This is nothing more than a hint to the sender so that it can retransmit a frame right away without waiting for a timer to expire.
- **Timers:** One problem that simple ACK/NACK schemes fail to address is recovering from a frame that is lost? Retransmission timers are used to resend frames that don't produce an ACK. When sending a frame, schedule a timer to expire at some time after the ACK should have been returned. If the timer goes o, retransmit the frame.

- **Sequence Numbers:** Retransmissions introduce the possibility of duplicate frames. To suppress duplicates, add sequence numbers to each frame, so tha1t 9a receiver can distinguish between new frames and old copies.

FLOW CONTROL:

Flow control deals with controlling the speed of the sender to match that of the receiver.

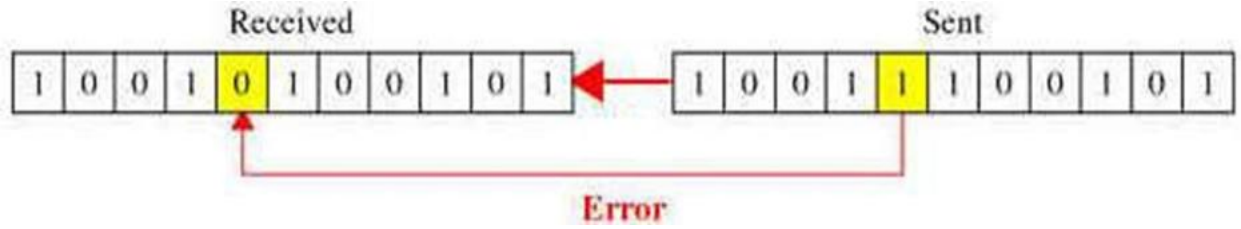
Two Approaches:

- **feedback-based flow control**, the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing
- **rate-based flow control**, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

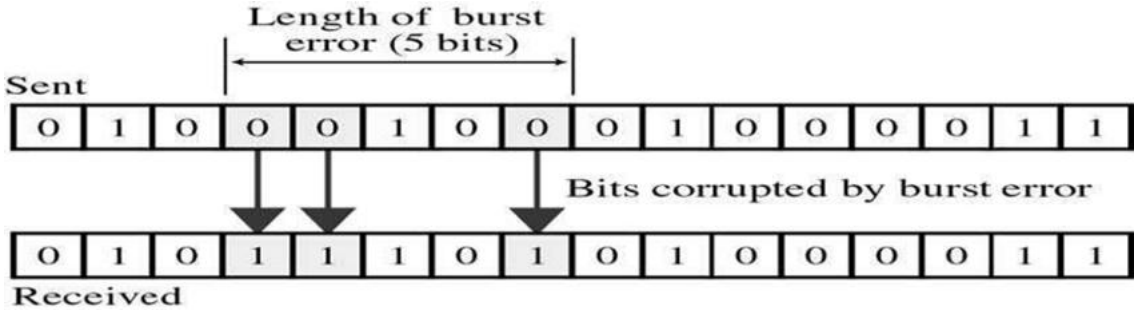
TYPES OF ERRORS:

There are two main types of errors in transmissions:

1. **Single bit error:** It means only one bit of data unit is changed from 1 to 0 or from 0 to 1.



2. **Burst error:** It means two or more bits in data unit are changed from 1 to 0 from 0 to 1. In burst error, it is not necessary that only consecutive bits are changed. The length of burst error is measured from first changed bit to last changed bit

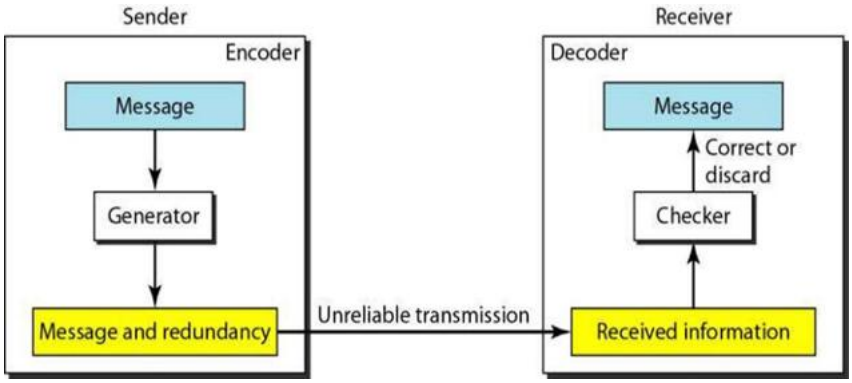


ERROR DETECTION AND CORRECTION:

Network designers have developed two basic strategies for dealing with errors. Both add redundant information to the data that is sent. One strategy is to include enough redundant information to enable the receiver to deduce what the transmitted data must have been. The other is to include only enough redundancy to allow the receiver to deduce that an error has occurred and have it request a retransmission. The former strategy uses **error-correcting codes** and the latter uses **error-detecting codes**.

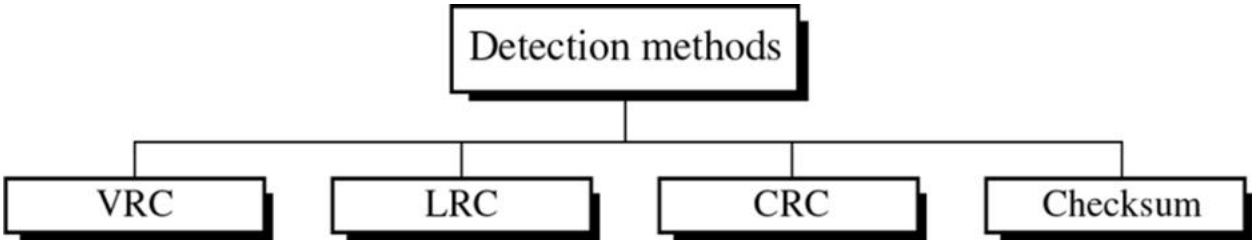
Error Detecting Codes: Include enough redundancy bits to *detect* errors and use ACKs and retransmissions to recover from the errors.

Error Correcting Codes: Include enough redundancy to detect and correct errors.



ERROR-DETECTING CODES:

Error detection means to decide whether the received data is correct or not without having a copy of the original message. Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.



1. **Vertical Redundancy Check(VRC):**

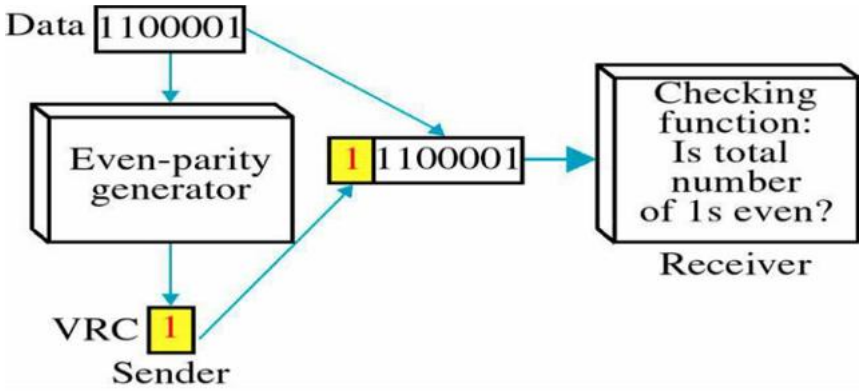
Append a single bit at the end of data block such that the number of one's is even

→À Even Parity (odd parity is similar)

0110011 →01100110

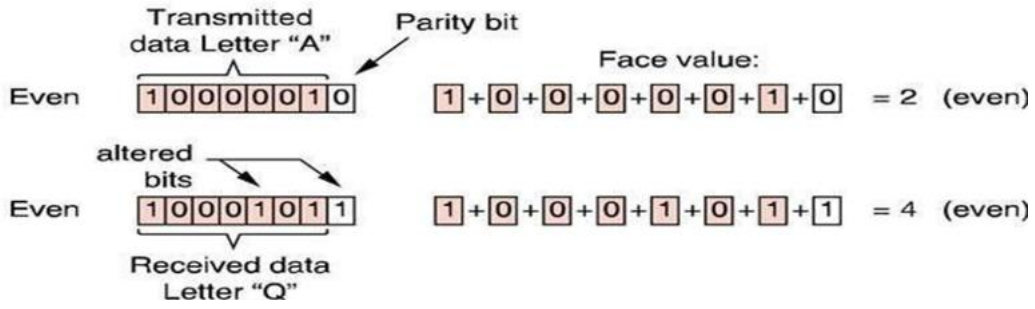
0110001 → 01100011

VRC is also known as **Parity Check**. Detects all odd-number errors in a data block.



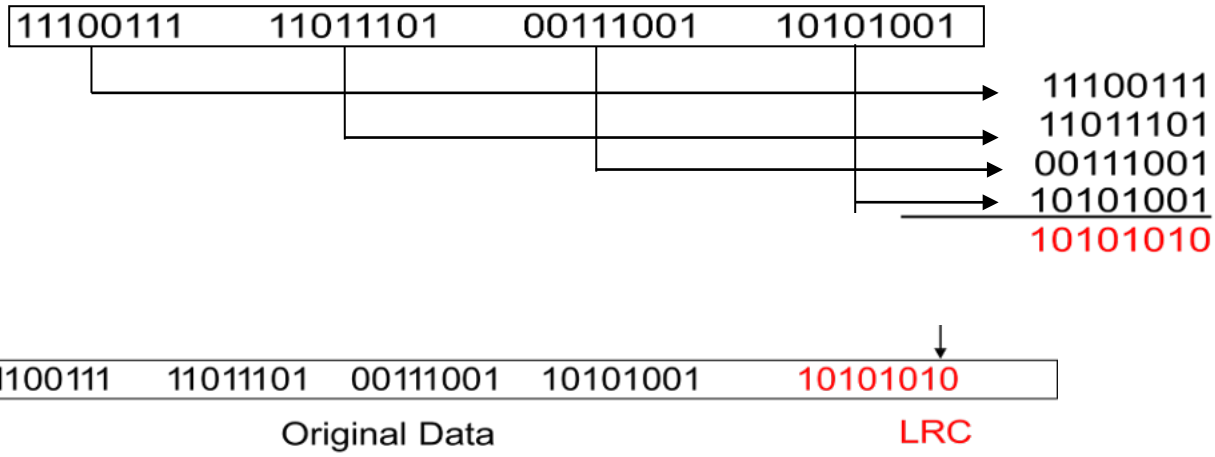
The problem with parity is that it can only detect odd numbers of bit substitution errors, i.e. 1 bit, 3bit, 5, bit, etc. Errors. If there two, four, six, etc. bits which are transmitted in error, using VRC will not be able to detect the error.

Multiple bit errors/even parity



2. Longitudinal Redundancy Check(LRC):

- Longitudinal Redundancy Checks (LRC) seek to overcome the weakness of simple, bit-oriented, one-directional parity checking.
- LRC adds a new character (instead of a bit) called the Block Check Character (BCC) to each block of data. Its determined like parity, but counted longitudinally through the message (also vertically)
- Its has better performance over VRC as it detects 98% of the burst errors (>10 errors) but less capable of detecting single errors



3. Cyclic Redundancy Check(CRC):

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. The CRC (**Cyclic Redundancy Check**), also known as a **polynomial code**

Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients 1, 1, 0, 0, 0, and 1: **1x5 + 1x4 + 0x3 + 0x2 + 0x1 + 1x0.**

Polynomial arithmetic is done modulo 2, according to the rules of algebraic field theory. It does not have carries for addition or borrows for subtraction. Both addition and subtraction are identical to **exclusive OR.** For example:

10011011	**00110011**	**11110000**	**01010101**
+ 11001010	**+ 11001101**	**− 10100110**	**− 10101111**
01010001	**11111110**	**01010110**	**11111010**

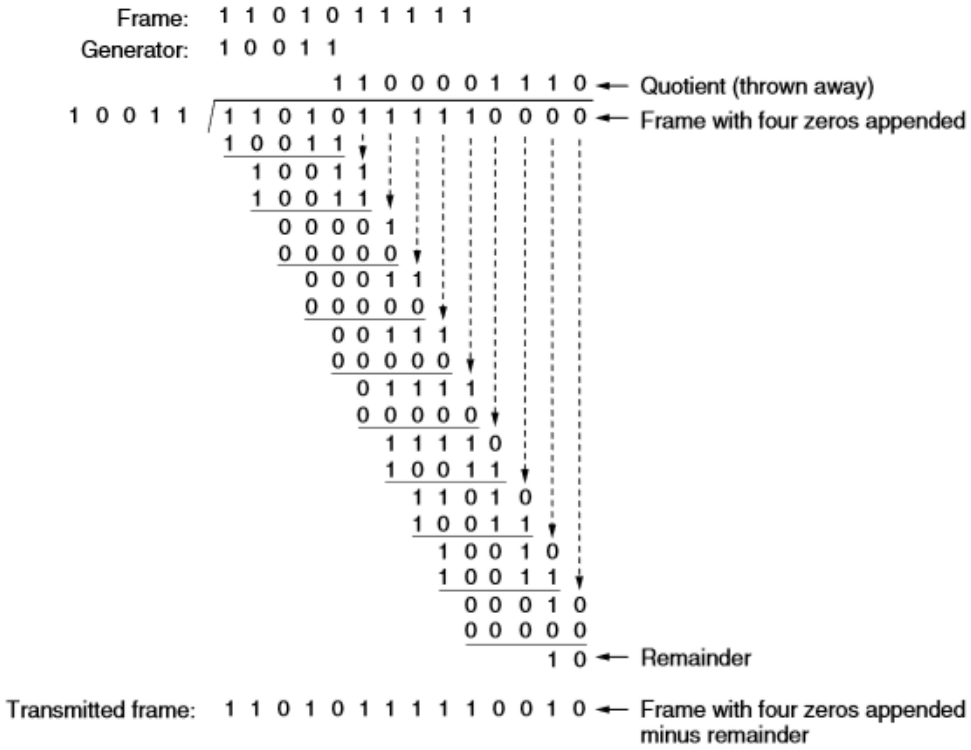
When the polynomial code method is employed, the sender and receiver must agree upon a **generator polynomial, G(x),** in advance. Both the high- and low order bits of the generator must be 1. To compute the CRC for some frame with m bits corresponding to the polynomial M(x), the frame must be longer than the generator polynomial. The idea is to append a CRC to the end of the frame in such a way that the polynomial represented by the check summed frame is divisible by G(x). When the receiver gets the check summed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error.

The **algorithm for computing the CRC is as follows:**

1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m + r bits and corresponds to the polynomial $x^rM(x)$.
2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^rM(x)$, using modulo 2 divisions.

3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtractions. The result is the check summed frame to be transmitted. Call its polynomial T(x).

Below figure illustrates the calculation for a frame 1101011111 using the generator $G(x) = x4 + x + 1$.



Example calculation of the CRC.

It should be clear that T(x) is divisible (modulo 2) by G(x). In any division problem, if you diminish the dividend by the remainder, what is left over is divisible by the divisor.

Example of CRC:

- Consider a message 110010 represented by the polynomial $M(x) = x^5 + x^4 + x$
  Consider a *generating polynomial* $G(x) = x^3 + x^2 + 1$ (1101)

This is used to generate a 3 bit CRC = C(x) to be appended to M(x).

Steps:

1. Multiply M(x) by x^3 (highest power in G(x)). i.e. Add 3 zeros. 110010000
2. Divide the result by G(x). The remainder = C(x).

1101 long division into 110010000 (with subtraction mod 2)

= 100100 remainder **100**

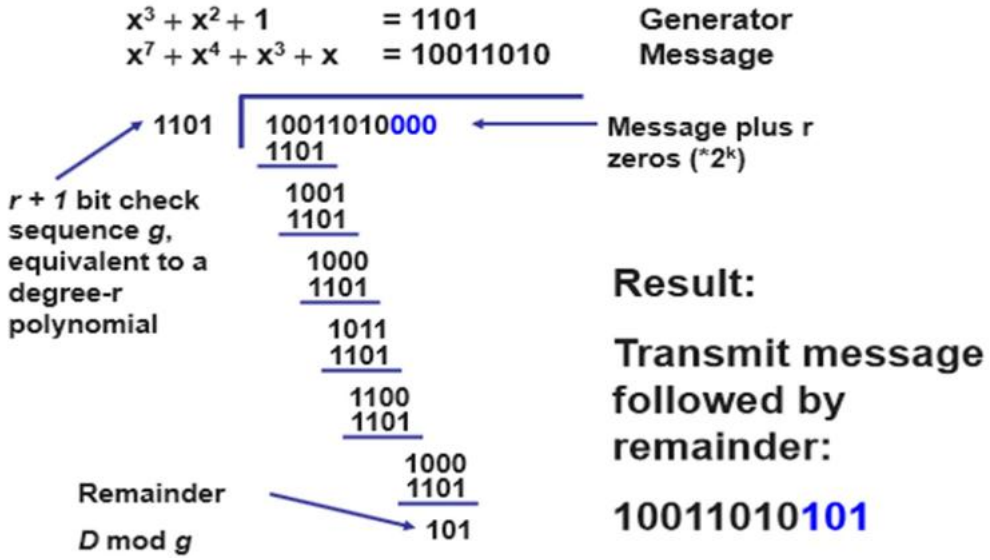
3. Transmit 110010000 + 100

To be precise, transmit: $T(x) = x^3M(x) + C(x)$

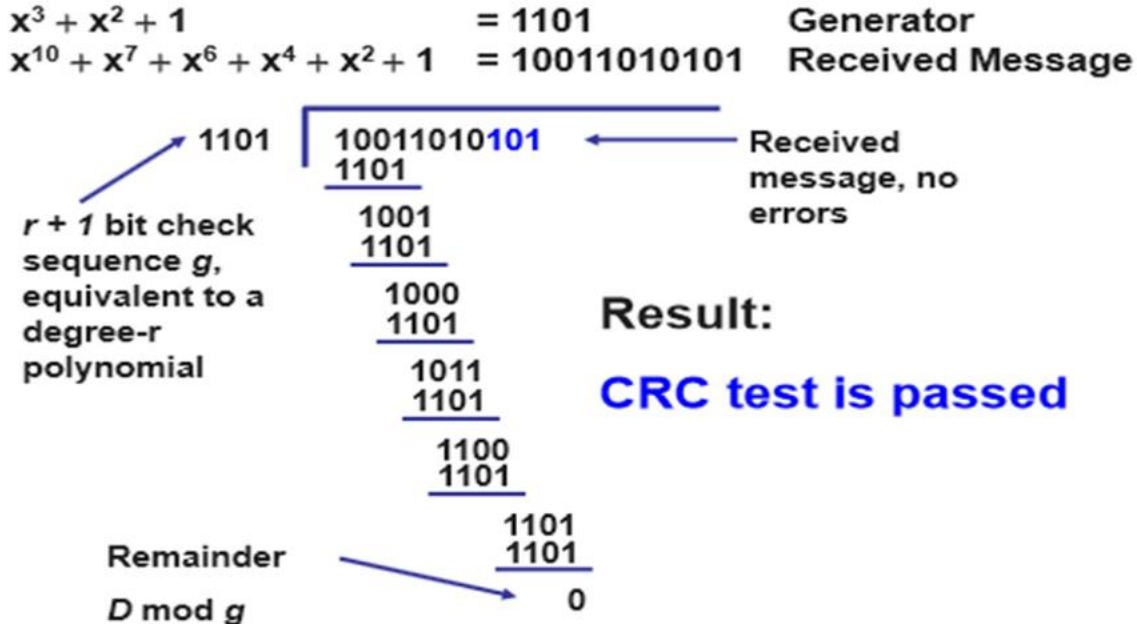
= 110010100

4. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

At sender side calculation of CRC:

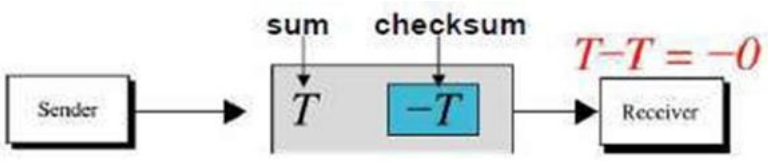


At receiver side calculation of CRC:



4. CHECKSUM:

- Checksum is the error detection scheme used in IP, TCP & UDP.
- Here, the data is divided into k segments each of n bits. In 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
- 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
- The checksum detects all errors involving an odd number of bits. It also detects most errors involving even number of bits.



Checksum procedure at sender and receiver end:

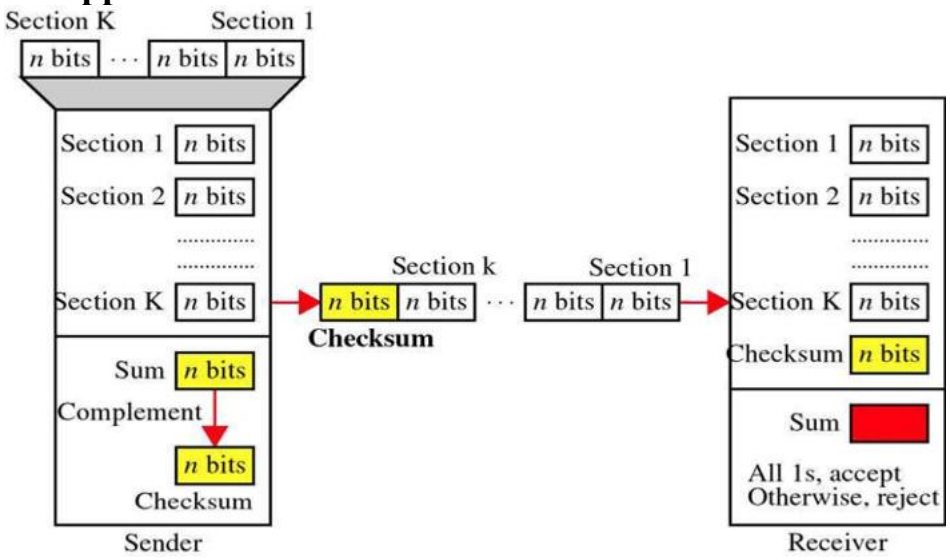
Sender:

- data is divided into k sections each n bits long
- all sections are added using 1-s complement to get the sum
- the sum is bit-wise complemented and becomes the checksum
- the checksum is sent with the data

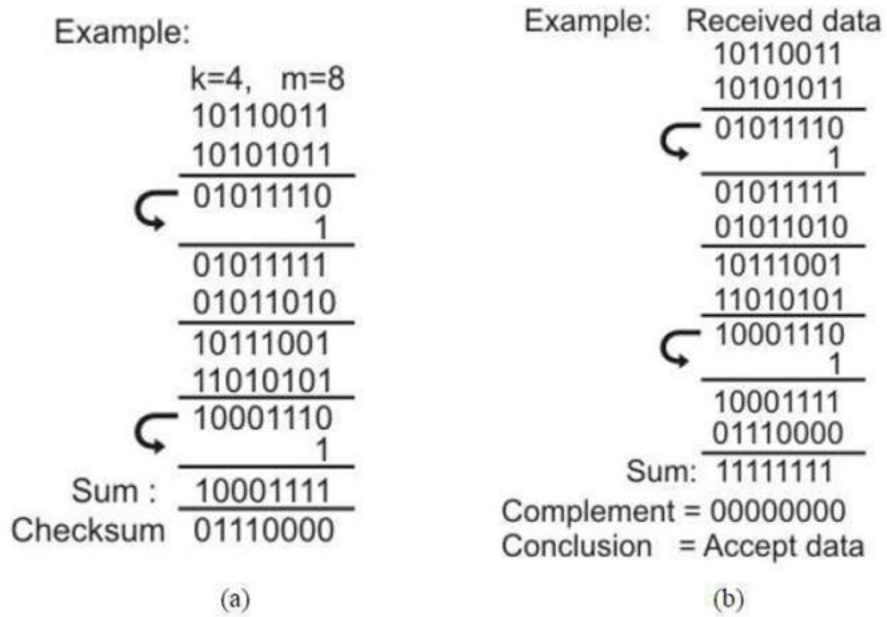
Receiver:

- data is divided into k sections each n bits long
- all sections are added using 1-s complement to get the sum
- the sum is bit-wise complemented
- if the result is zero, the data is accepted, otherwise it is rejected

Diagrammatic approach:



Example for Checksum:

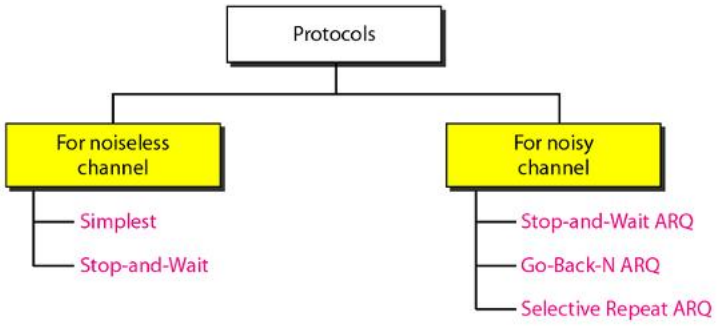


(a) Sender's end for the calculation of the checksum. (b) Receiving end for checking the checksum

Elementary Data Link Layer protocols:

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages.

We divide the discussion of protocols into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels. The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels.



NOISELESS CHANNELS:

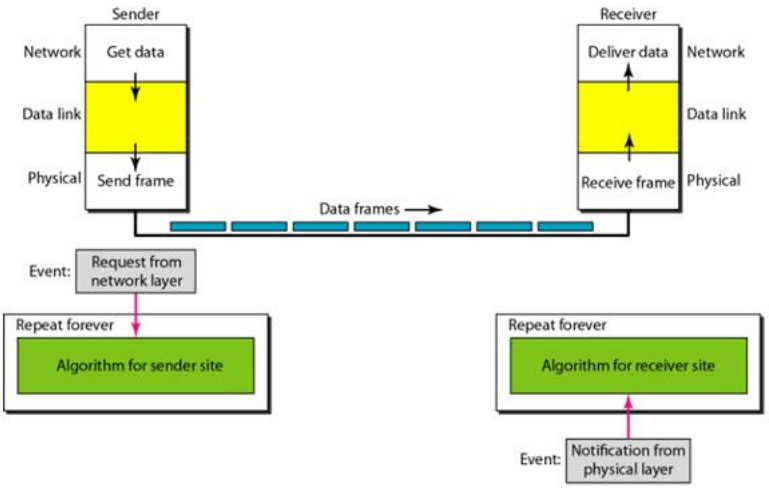
Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel. The first is a protocol that does not use flow control; the second is the one that does. Of course, neither has error control because we have assumed that the channel is a perfect noiseless channel.

Simplest Protocol:

Our first protocol, which we call the Simplest Protocol for lack of any other name, is one that has no flow or error control. Like other protocols we will discuss in this chapter, it is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver. We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.

Design

There is no need for flow control in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer. The data link layers of the sender and receiver provide transmission services for their network layers.



Algorithms

Sender-site algorithm for the simplest protocol

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(RequestToSend))           //There is a packet to send
5    {
6       GetData();
7       MakeFrame();
8       SendFrame();                    //Send the frame
9    }
10 }
```

Analysis The algorithm has an infinite loop, which means lines 3 to 9 are repeated forever once the program starts. The algorithm is an event-driven one, which means that it *sleeps* (line 3) until an event *wakes* it *up* (line 4). This means that there may be an undefined span of time between the execution of line 3 and line 4; there is a gap between these actions. When the event, a request from the network layer, occurs, lines 6 through 8 are executed. The program then repeats The loop and again sleeps at line 3 until the next occurrence of the event. We have written pseudo code for the main process. We do not show any details for the modules Get Data, Make Frame, and Send Frame.

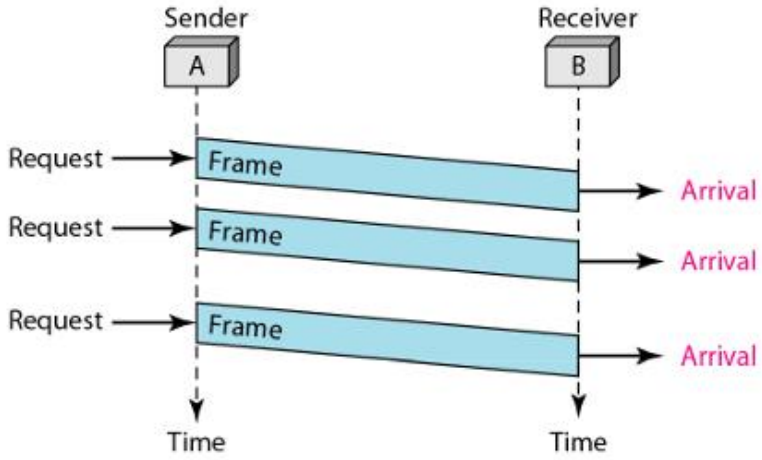
Receiver-site algorithm for the simplest protocol

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData();                  //Deliver data to network layer
9    }
10 }
```

Analysis This algorithm has the same format as above Algorithm except that the direction of the frames and data is upward. The event here is the arrival of a data frame. After the event occurs, the data link layer receives the frame from the physical layer using the ReceiveFrame() process, extracts the data from the frame using the ExtractData() process, and delivers the data to the network layer using the DeliverData() process. Here, we also have an event-driven algorithm because the algorithm never knows when the data frame will arrive.

Example:

Below Figure shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.



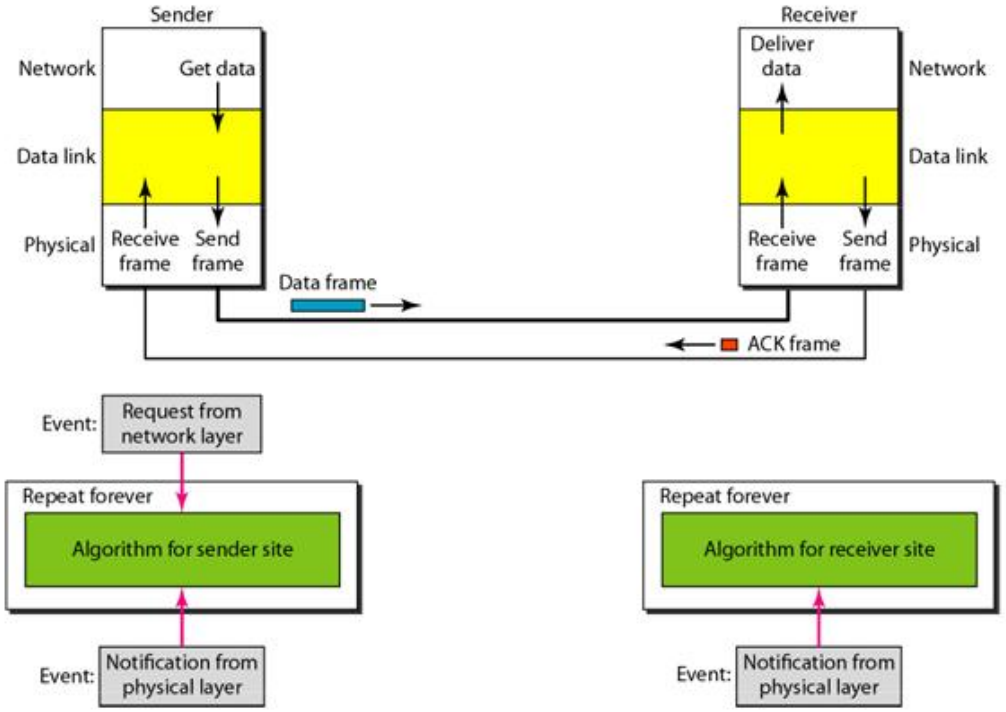
Stop-and-Wait Protocol:

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

The protocol we discuss now is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

Design

We can see the traffic on the forward channel (from sender to receiver) and the reverse channel. At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. We therefore need a half-duplex link.



Algorithms

Sender-site algorithm for Stop-and- Wait Protocol

```
1  while(true)                          //Repeat forever
2  canSend = true                       //Allow the first frame to go
3  {
4   WaitForEvent();                     // Sleep until an event occurs
5   if(Event(RequestToSend) AND canSend)
6   {
7      GetData();
8      MakeFrame();
9      SendFrame();                     //Send the data frame
10     canSend = false;                 //Cannot send until ACK arrives
11  }
12  WaitForEvent();                     // Sleep until an event occurs
13  if(Event(ArrivalNotification)       // An ACK has arrived
14   {
15     ReceiveFrame();                  //Receive the ACK frame
16     canSend = true;
17   }
18 }
```

Analysis Here two events can occur: a request from the network layer or an arrival notification from the physical layer. The responses to these events must alternate. In other words, after a frame is sent, the algorithm must ignore another network layer request until that frame is acknowledged. We know that two arrival events cannot happen one after another because the channel is error-free and does not duplicate the frames. The requests from the network layer, however, may happen one after another without an arrival event in between. We need somehow to prevent the immediate sending of the data frame. Although there are several methods, we have used a simple *canSend* variable that can either be true or false. When a frame is sent, the variable is set to false to indicate that a new network request cannot be sent until *canSend* is true. When an ACK is received, canSend is set to true to allow the sending of the next frame.

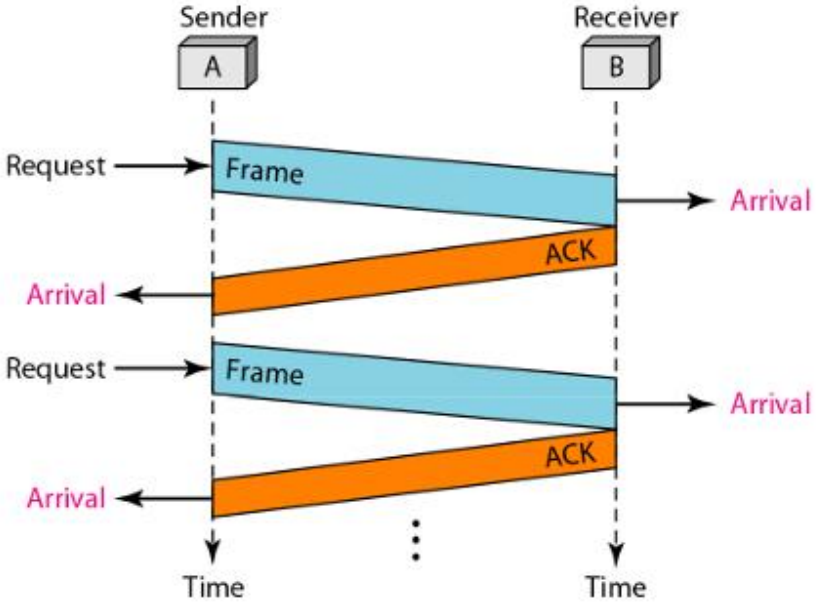
Receiver-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                          //Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrives
5    {
6       ReceiveFrame();
7       ExtractData();
8       Deliver(data);                  //Deliver data to network layer
9       SendFrame();                    //Send an ACK frame
10   }
11 }
```

Analysis This is very similar to above Algorithm with one exception. After the data frame arrives, the receiver sends an ACK frame (line 9) to acknowledge the receipt and allow the sender to send the next frame.

Example:

Below Figure shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.



NOISY CHANNELS:

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request:

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.

The lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, 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, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Sequence Numbers

As we discussed, the protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.

One important consideration is the range of the sequence numbers. Since we want to minimize the frame size, we look for the smallest range that provides unambiguous communication. The sequence numbers of course can wrap around. For example, if we decide that the field is *m* bits long, the sequence numbers start from 0, go to 2^m - 1, and then are repeated.

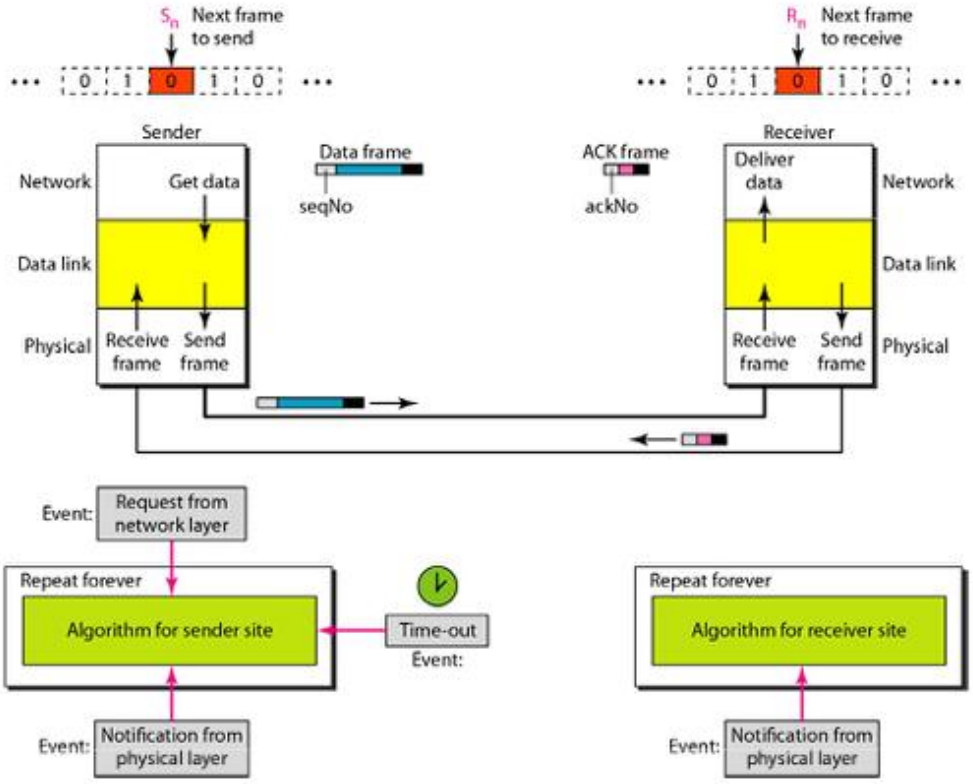
Acknowledgment Numbers

Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

Design

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

The receiver has a control variable, which we call *Rn* (receiver, next frame expected), that holds the number of the next frame expected. When a frame is sent, the value of *Sn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of *Rn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable *Sn* points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; *Rn* points to the slot that matches the sequence number of the expected frame.



Algorithms

Sender-site algorithm for Stop-and- Wait ARQ

```
1  Sn = 0;                          // Frame 0 should be sent first
2  canSend = true;                  // Allow the first request to go
3  while(true)                      // Repeat forever
4  {
5    WaitForEvent();                // Sleep until an event occurs
6    if(Event(RequestToSend) AND canSend)
7    {
8       GetData();
9       MakeFrame(Sn);                    //The seqNo is Sn
10      StoreFrame(Sn);                   //Keep copy
11      SendFrame(Sn);
12      StartTimer();
13      Sn = Sn + 1;
14      canSend = false;
15   }
16   WaitForEvent();                      // Sleep
17      if(Event(ArrivalNotification)     // An ACK has arrived
18      {
19        ReceiveFrame(ackNo);            //Receive the ACK frame
20        if(not corrupted AND ackNo == Sn) //Valid ACK
21          {
22            Stoptimer();
23            PurgeFrame(Sn-1);           //Copy is not needed
24            canSend = true;
25          }
26       }
27
28      if(Event(TimeOut)                 // The timer expired
29      {
30       StartTimer();
31       ResendFrame(Sn-1);               //Resend a copy check
32      }
33 }
```

Analysis We first notice the presence of *Sn'* the sequence number of the next frame to be sent. This variable is initialized once (line 1), but it is incremented every time a frame is sent (line 13) in preparation for the next frame. However, since this is modulo-2 arithmetic, the sequence numbers are 0, 1, 0, 1, and so on. Note that the processes in the first event (SendFrame, Store Frame, and Purge Frame) use an *Sn* defining the frame sent out. We need at least one buffer to hold this frame until we are sure that it is received safe and sound. Line 10 shows that before the frame is sent, it is stored. The copy is used for resending a corrupt or lost frame. We are still using the canSend variable to prevent the network layer from making a request before the previous frame is received safe and sound. If the frame is not corrupted and the ackNo of theACK frame matches the sequence number of the next frame to send, we stop the timer and purge the copy of the data frame we saved. Otherwise, we just ignore this event and wait for the next event to happen. After each frame is sent, a timer is started. When the timer expires (line 28), the frame is resent and the timer is restarted.

Receiver-site algorithm for Stop-and-WaitARQ Protocol

```
1  Rn = 0;                        // Frame 0 expected to arrive first
2  while(true)
3  {
4    WaitForEvent();              // Sleep until an event occurs
5    if(Event(ArrivalNotification))  //Data frame arrives
6    {
7       ReceiveFrame();
8       if(corrupted(frame));
9          sleep();
10      if(seqNo == Rn)              //Valid data frame
11      {
12       ExtractData();
13        DeliverData();             //Deliver data
14        Rn = Rn + 1;
15      }
16       SendFrame(Rn);              //Send an ACK
17   }
18 }
```

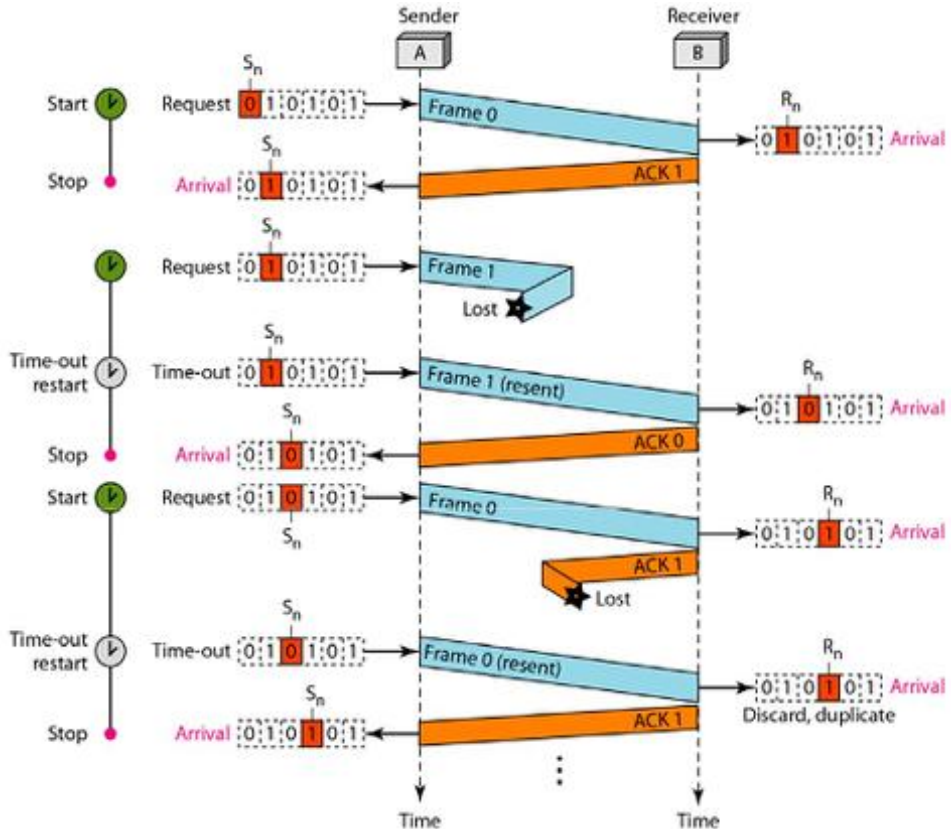
Analysis This is noticeably different from Algorithm 11.4. First, all arrived data frames that are corrupted are ignored. If the seqNo of the frame is the one that is expected *(Rn),* the frame is accepted, the data are delivered to the network layer, and the value of *Rn* is incremented. However, there is one subtle point here. 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, however, just 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.

Efficiency

The Stop-and-Wait ARQ discussed in the previous section is very inefficient if our channel is *thick* and *long.* By *thick,* we mean that our channel has a large bandwidth; by *long,* we mean the round-trip delay is long. The product of these two is called the bandwidth delay product, as we discussed in Chapter 3. We can think of the channel as a pipe. The bandwidth-delay product then is the volume of the pipe in bits. The pipe is always there. If we do not use it, we are inefficient. The bandwidth-delay product is a measure of the number of bits we can send out of our system while waiting for news from the receiver.

Example

Below Figure shows an example of Stop-and-Wait ARQ. Frame **a** 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 **a** 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, it resends frame 0, which is acknowledged.



Pipelining

In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining. There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent. However, pipelining does apply to our next two protocols because several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-N Automatic Repeat Request:

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

Go-Back-N Automatic Repeat Request protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows *m* bits for the sequence number, the sequence numbers range from 0 to *2m* - 1. For example, if *m* is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

In other words, the sequence numbers are modulo-2^m

.

Sliding Window

In this protocol the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window.

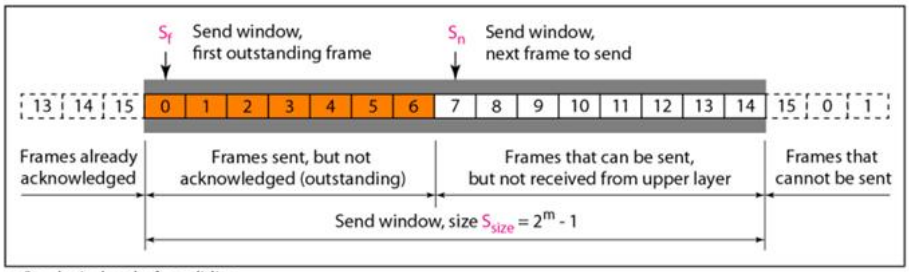
The send window is an imaginary box covering the sequence numbers of the data frames which can be in transmit. In each window position, some of these sequence numbers define the frames that have been sent;

others define those that can be sent. The maximum size of the window is 2^m - 1 we let the size be fixed and set to the maximum value, below figure **a** shows a sliding window of size 15 *(m* =4).

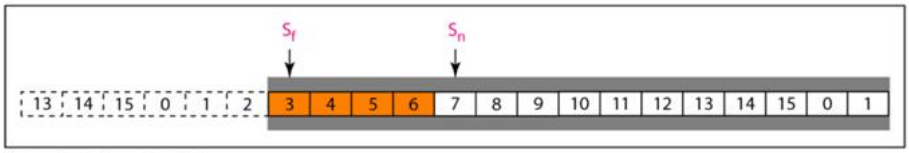
The window at any time divides the possible sequence numbers into four regions. The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region, colored in Figure **a**, 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. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

Below Figure **b** shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. As we will see shortly, the acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure **b**, frames 0, I, and 2 are acknowledged, so the window has slid to the right three slots. Note that the value of *Sf* is 3 because frame 3 is now the first outstanding frame.

The window itself is an abstraction; three variables define its size and location at any time. We call these variables *Sf(send* window, the first outstanding frame), *Sn* (send window, the next frame to be sent), and Ssize (send window, size). The variable *Sf* defines the sequence number of the first (oldest) outstanding frame. The variable *Sn* holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window, which is fixed in our protocol.



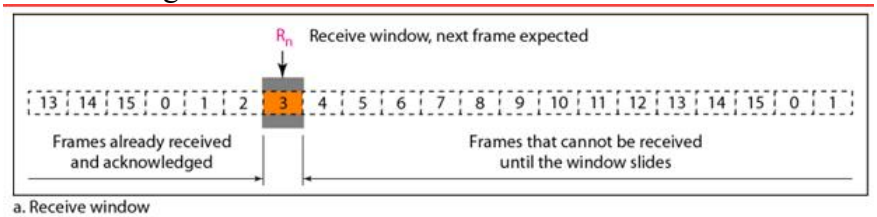
a. Send window before sliding



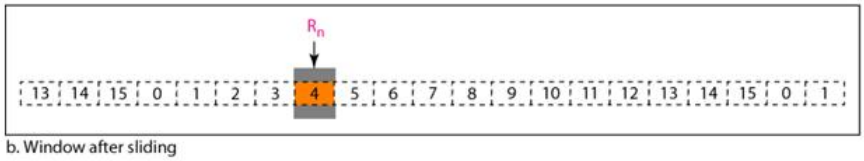
b. Send window after sliding

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. Below figure shows the receive window.

We need only one variable *Rn* (receive window, next frame expected) to define this abstraction. The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of *Rn* is accepted and acknowledged.



a. Receive window



b. Window after sliding

Timers

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

Acknowledgment

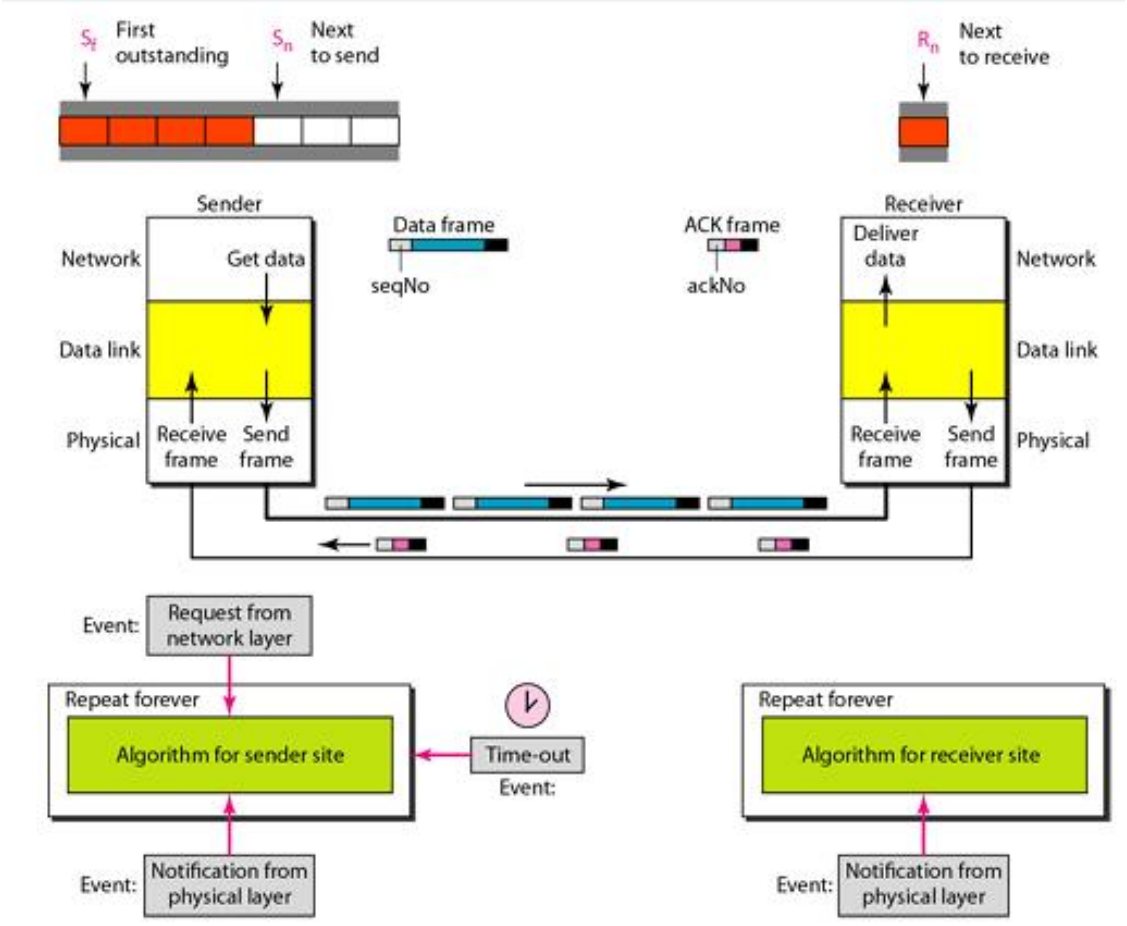
The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This, in turn, 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.

Resending a Frame

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

Design

Below Figure shows the design for this protocol. As we can see, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction. The idea is similar to Stop-and-Wait ARQ; the difference is that the send window allows us to have as many frames in transition as there are slots in the send window.



Algorithms
Go-Back-N sender algorithm

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend))            //A packet to send
9    {
10      if(Sn-Sf >= Sw)                  //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      if(timer not running)
18           StartTimer();
19   }
20
21   if(Event(ArrivalNotification)) //ACK arrives
22   {
23      Receive(ACK);
24      if(corrupted(ACK))
25           Sleep();
26      if((ackNo>Sf)&&(ackNo<=Sn))  //If a valid ACK
27      While(Sf <= ackNo)
28       {
29        PurgeFrame(Sf);
30        Sf = Sf + 1;
31       }
32       StopTimer();
33   }
34
35   if(Event(TimeOut))                  //The timer expires
36   {
37    StartTimer();
38    Temp = Sf;
39    while(Temp < Sn);
40     {
41      SendFrame(Sf);
42      Sf = Sf + 1;
43     }
44   }
45 }
```

Analysis this algorithm first initializes three variables. Unlike Stop-and-Wait ARQ, this protocol allows several requests from the network layer without the need for other events to occur; we just need to be sure that the window is not full (line 12). In our approach, if the window is full, the request is just ignored and the network layer needs to try again. Some implementations use other methods such as enabling or disabling the network layer. The handling of the arrival event is more complex than in the previous protocol. If we receive a corrupted ACK, we ignore it.

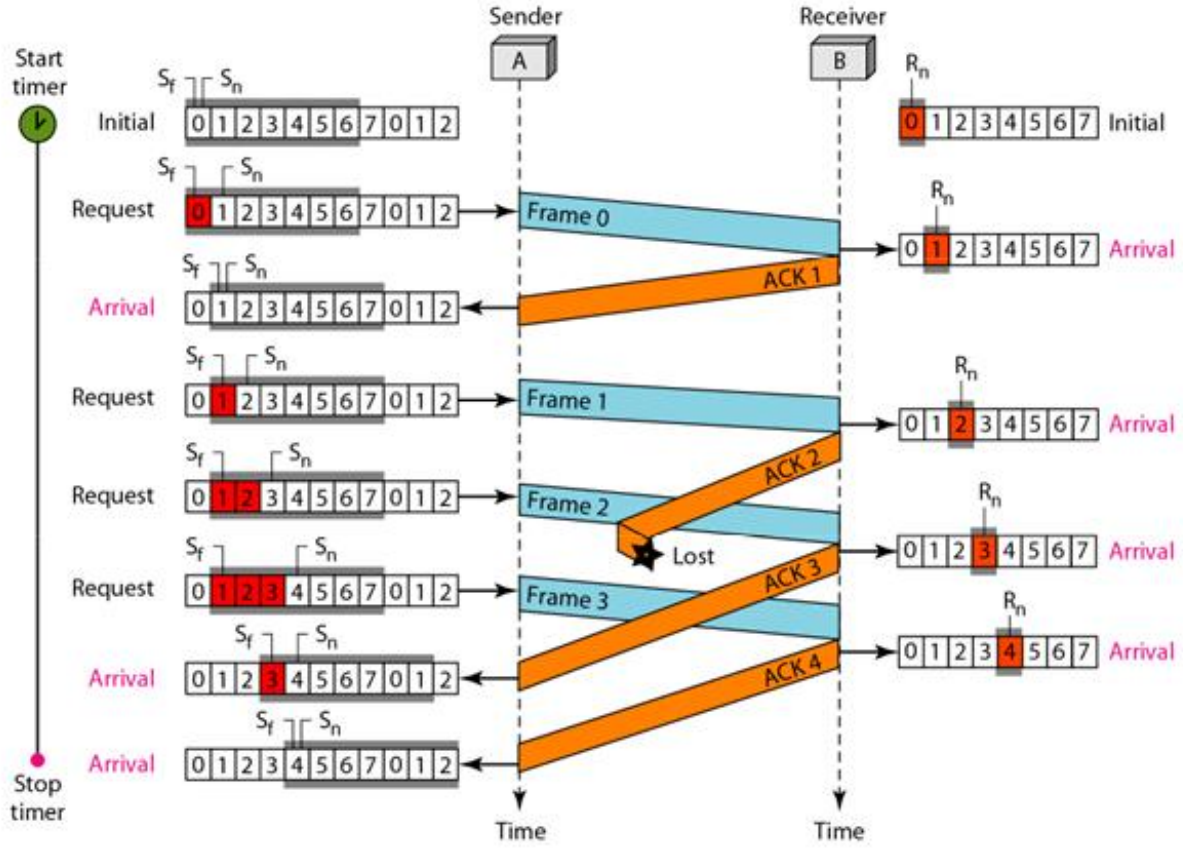
```
1  Rn = 0;
2
3  while (true)                          //Repeat forever
4  {
5    WaitForEvent();
6
7    if(Event(ArrivalNotification)) /Data frame arrives
8    {
9       Receive(Frame);
10      if(corrupted(Frame))
11           Sleep();
12      if(seqNo == Rn)                  //If expected frame
13      {
14        DeliverData();                 //Deliver data
15        Rn = Rn + 1;                   //Slide window
16        SendACK(Rn);
17      }
18   }
19 }
```

Analysis This algorithm is simple. We ignore a corrupt or out-of-order frame. If a frame arrives with an expected sequence number, we deliver the data, update the value of *Rn,* and send an ACK with the ackNa showing the next frame expected.

Example

Below Figure shows an example of Go-Back-*N.* This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.

After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK3.

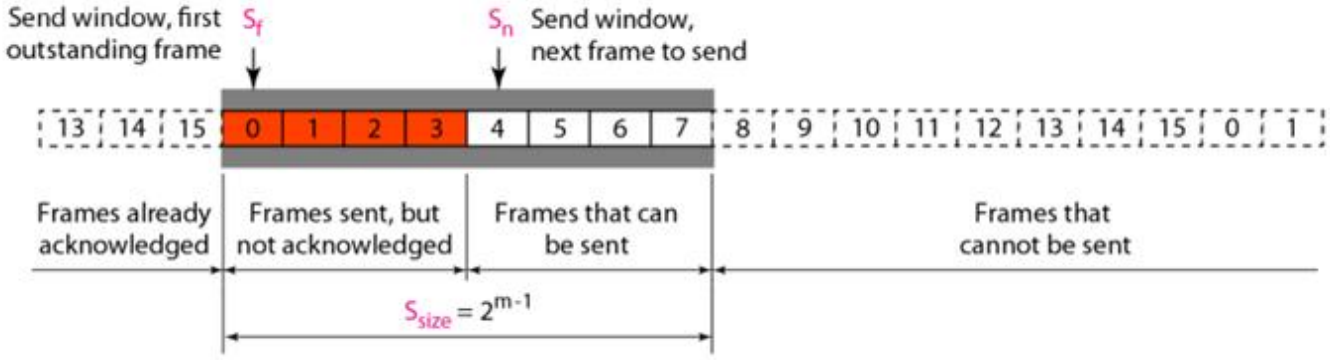


Selective Repeat Automatic Repeat Request:

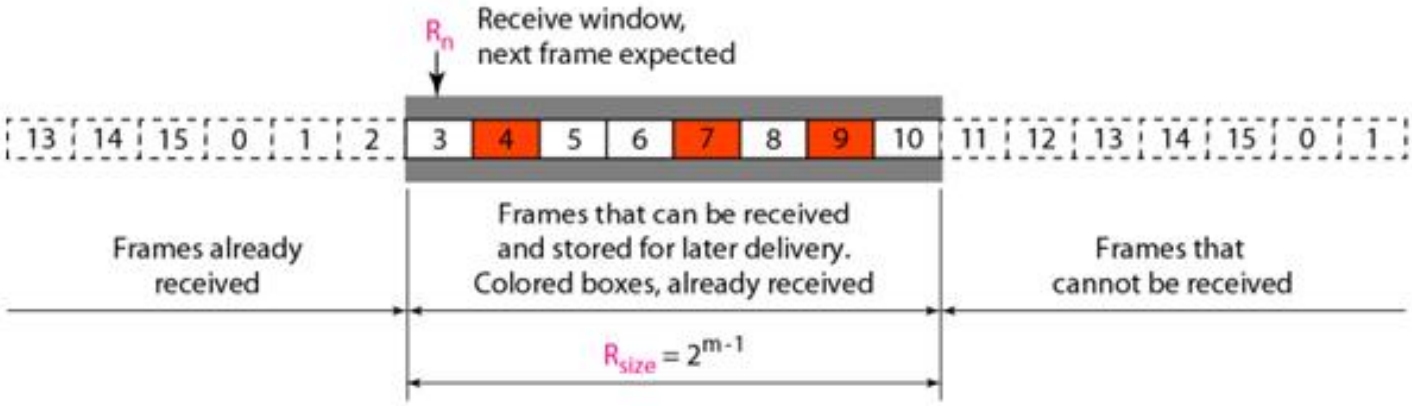
Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage,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

Windows

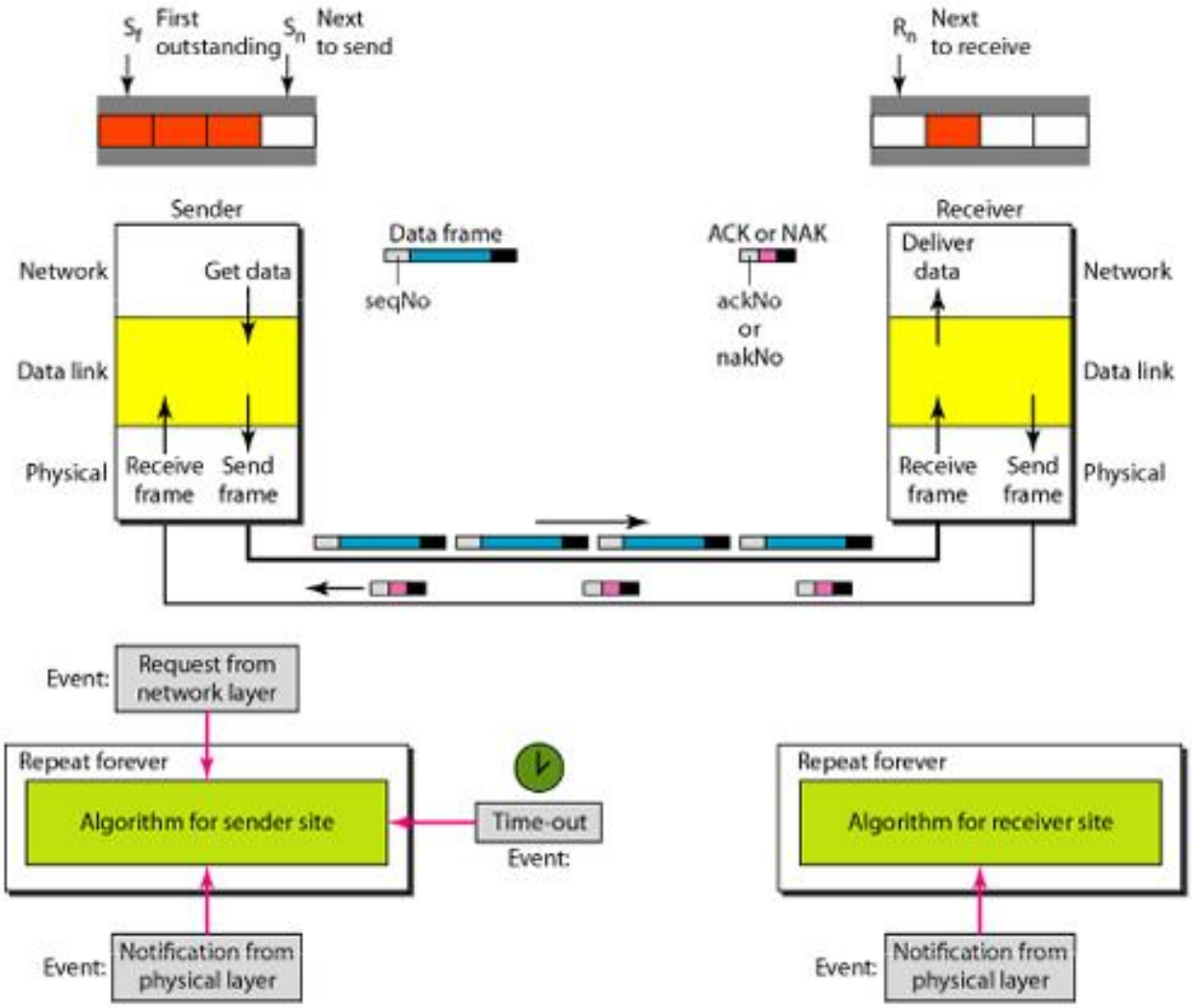
The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are differences between the windows in this protocol and the ones in Go-Back-N. First, the size of the send window is much smaller; it is $2^m - 1$. The reason for this will be discussed later. Second, the receive window is the same size as the send window. The send window maximum size can be $2^m - 1$. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the size of the window is just 8



The receive window in Selective Repeat is totally different from the one in GoBack-N. First, the size of the receive window is the same as the size of the send window $(2^m - 1)$. 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.



Design



Algorithms

Sender-side Selective Repeat algorithm

```
1  Sw = 2^(m-1) ;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7    WaitForEvent();
8    if(Event(RequestToSend))            //There is a packet to send
9    {
10     if(Sn-Sf >= Sw)                   //If window is full
11           Sleep();
12     GetData();
13     MakeFrame(Sn);
14     StoreFrame(Sn);
15     SendFrame(Sn);
16     Sn = Sn + 1;
17     StartTimer(Sn);
18   }
19
```

```
20   if(Event(ArrivalNotification)) //ACK arrives
21   {
22      Receive(frame);             //Receive ACK or NAK
23      if(corrupted(frame))
24           Sleep();
25      if (FrameType == NAK)
26         if (nakNo between Sf and Sn)
27         {
28          resend(nakNo);
29          StartTimer(nakNo);
30         }
31      if (FrameType == ACK)
32         if (ackNo between Sf and Sn)
33         {
34           while(sf < ackNo)
35           {
36            Purge(sf);
37            StopTimer(sf);
38            Sf = Sf + 1;
39           }
40         }
41   }
42
43   if(Event(TimeOut(t)))          //The timer expires
44   {
45    StartTimer(t);
46    SendFrame(t);
47   }
48 }
```

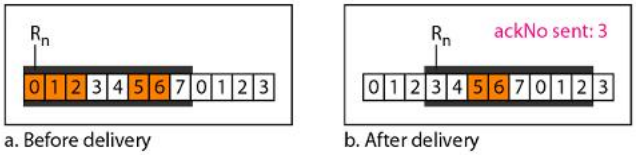
Analysis The handling of the request event is similar to that of the previous protocol except that one timer is started for each frame sent. The arrival event is more complicated here. An ACK or a NAK frame may arrive. If a valid NAK frame arrives, we just resend the corresponding frame. If a valid ACK arrives, we use a loop to purge the buffers, stop the corresponding timer. and move the left wall of the window. The time-out event is simpler here; only the frame which times out is resent.

Receiver-site Selective Repeat algorithm

```
1  Rn = 0;
2  NakSent = false;
3  AckNeeded = false;
4  Repeat(for all slots)
5      Marked(slot) = false;
6
7  while (true)                          //Repeat forever
8  {
9    WaitForEvent();
10
11   if(Event(ArrivalNotification))      /Data frame arrives
12   {
13      Receive(Frame);
14      if(corrupted(Frame))&& (NOT NakSent)
15      {
16       SendNAK(Rn);
17       NakSent = true;
18       Sleep();
19      }
20      if(seqNo <> Rn)&& (NOT NakSent)
21      {
22       SendNAK(Rn);
```

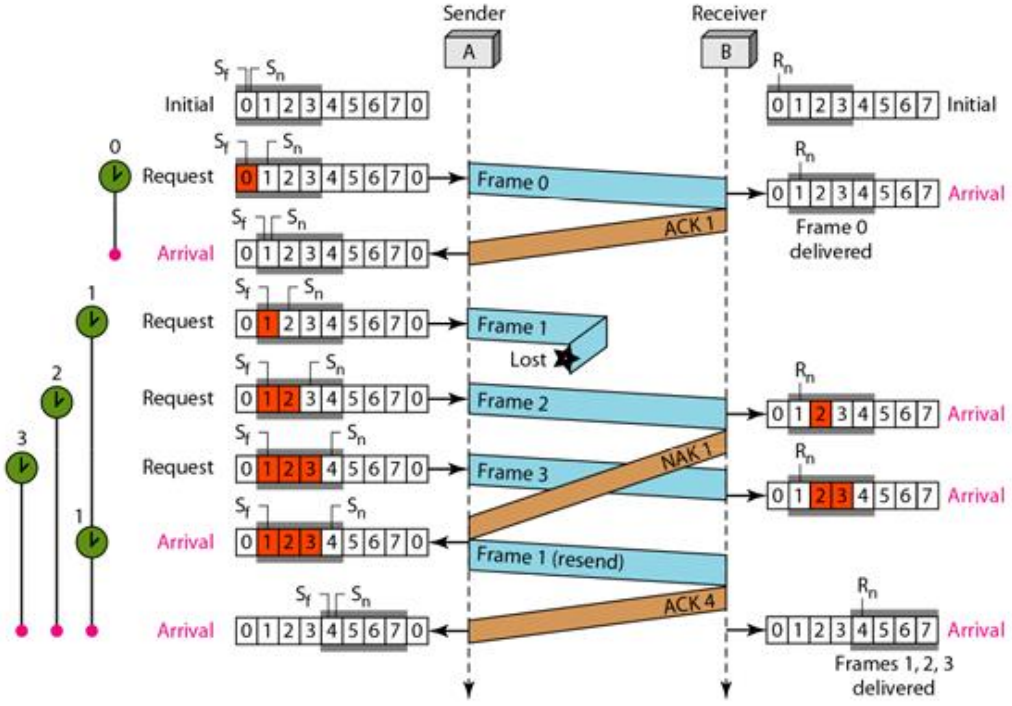
```
23        NakSent = true;
24        if ((seqNo in window)&&(!Marked(seqNo))
25        {
26         StoreFrame(seqNo)
27         Marked(seqNo)= true;
28         while(Marked(Rn))
29         {
30          DeliverData(Rn);
31          Purge(Rn);
32          Rn = Rn + 1;
33          AckNeeded = true;
34         }
35          if(AckNeeded);
36          {
37          SendAck(Rn);
38          AckNeeded = false;
39          NakSent = false;
40          }
41        }
42       }
43   }
44 }
```

Analysis Here we need more initialization. In order not to overwhelm the other side with NAKs, we use a variable called NakSent. To know when we need to send an ACK, we use a variable called AckNeeded. Both of these are initialized to false. We also use a set of variables to mark the slots in the receive window once the corresponding frame has arrived and is stored. If we receive a corrupted frame and a NAK has not yet been sent, we send a NAK to tell the other site that we have not received the frame we expected. If the frame is not corrupted and the sequence number is in the window, we store the frame and mark the slot. If contiguous frames, starting from *Rn* have been marked, we deliver their data to the network layer and slide the window. Below Figure shows this situation.



Example

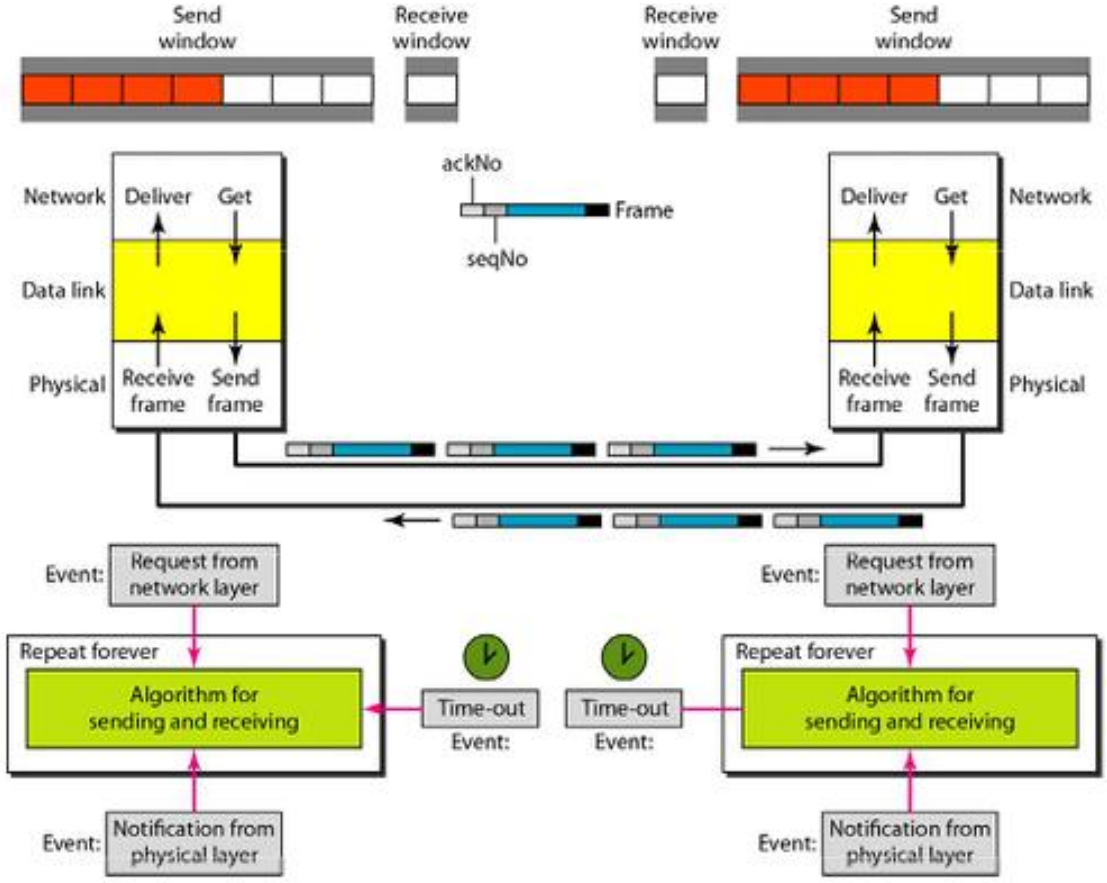
This example is similar to go back N Example in which frame 1 is lost. We show how Selective Repeat behaves in this case. Below Figure shows the situation.



Piggybacking:

The three protocols we discussed in this section are all unidirectional: 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. A technique called **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 about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

We show the design for a Go-Back-N ARQ using piggybacking in below Figure. Note that each node now has two windows: one send window and one receive window. Both also need to use a timer. Both are involved in three types of events: request, arrival, and time-out. However, the arrival event here is complicated; when a frame arrives, the site needs to handle control information as well as the frame itself. Both of these concerns must be taken care of in one event, the arrival event. The request event uses only the send window at each site; the arrival event needs to use both windows.



HDLC:

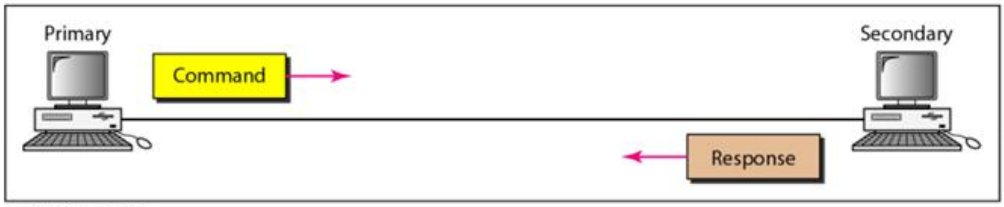
High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms

Configurations and Transfer Modes

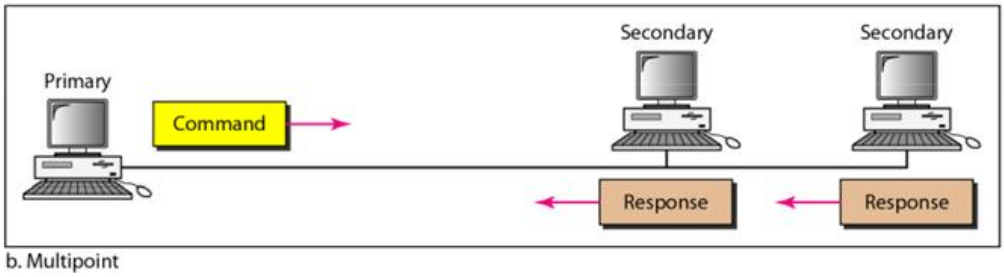
HDLC provides two common transfer modes that can be used in different configurations: normal response mode (NRM) and asynchronous balanced mode (ABM).

Normal Response Mode

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands; a secondary station can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in below Figure.



a. Point-to-point



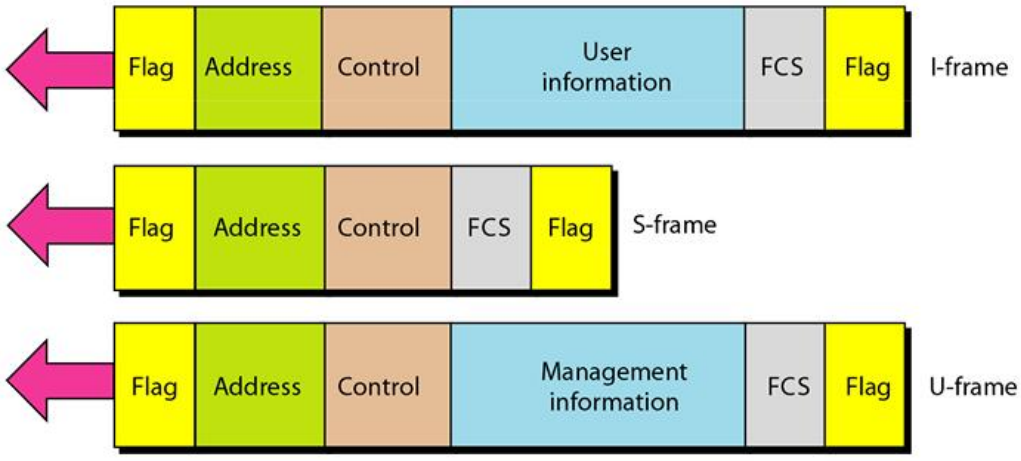
b. Multipoint

Asynchronous Balanced Mode

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in below Figure. This is the common mode today.

Frames

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: information frames (I-frames), supervisory frames (S-frames), and unnumbered frames (V-frames). Each type of frame serves as an envelope for the transmission of a different type of message. I-frames are used to transport user data and control information relating to user data (piggybacking). S-frames are used only to transport control information. V-frames are reserved for system management. Information carried by V-frames is intended for managing the link itself.



Fields

Let us now discuss the fields and their use in different frame types.

Flag field. The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.

Address field. The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary creates the frame, it contains *afrom* address. An address field can be 1 byte or several bytes long, depending on the needs of the network. One byte can identify up to 128 stations (l bit is used for another purpose). Larger networks require multiple-byte address

fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0; only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come.

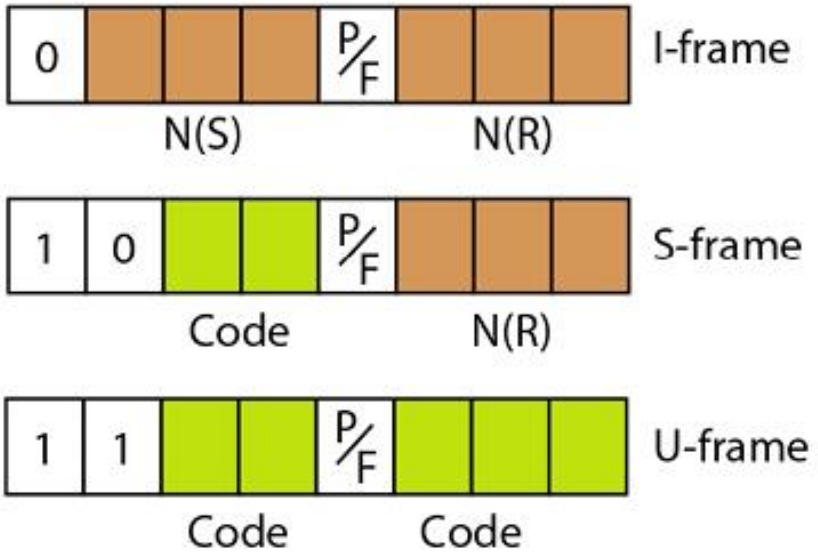
Control field. The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type. We discuss this field later and describe its format for each frame type.

Information field. The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.

FCS field. The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

Control Field

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in greater detail. The format is specific for the type of frame, as shown in below Figure.



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called *N(S),* define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7; but in the extension format, in which the control field is 2 bytes, this field is larger. The last 3 bits, called *N(R),* correspond to the acknowledgment number when piggybacking is used. The single bit between *N(S)* and *N(R)* is called the *PIF* bit. The *PIP* field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate (e.g., when the station either has no data of its own to send or needs to send a command or response other than an acknowledgment). S-frames do not have information fields. If the first 2 bits of the control field is 10, this means the frame is an S-frame. The last 3 bits, called *N(R),* corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame. The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

Receive ready (RR). If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value *N(R)* field defines the acknowledgment number.

Receive not ready (RNR). If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of *NCR)* is the acknowledgment number.
Reject (REJ). If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in *Go-Back-N* ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of *NCR)* is the negative acknowledgment number.
Selective reject (SREJ). If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat.* The value of *N(R)* is the negative acknowledgment number.

Control Field for V-Frames

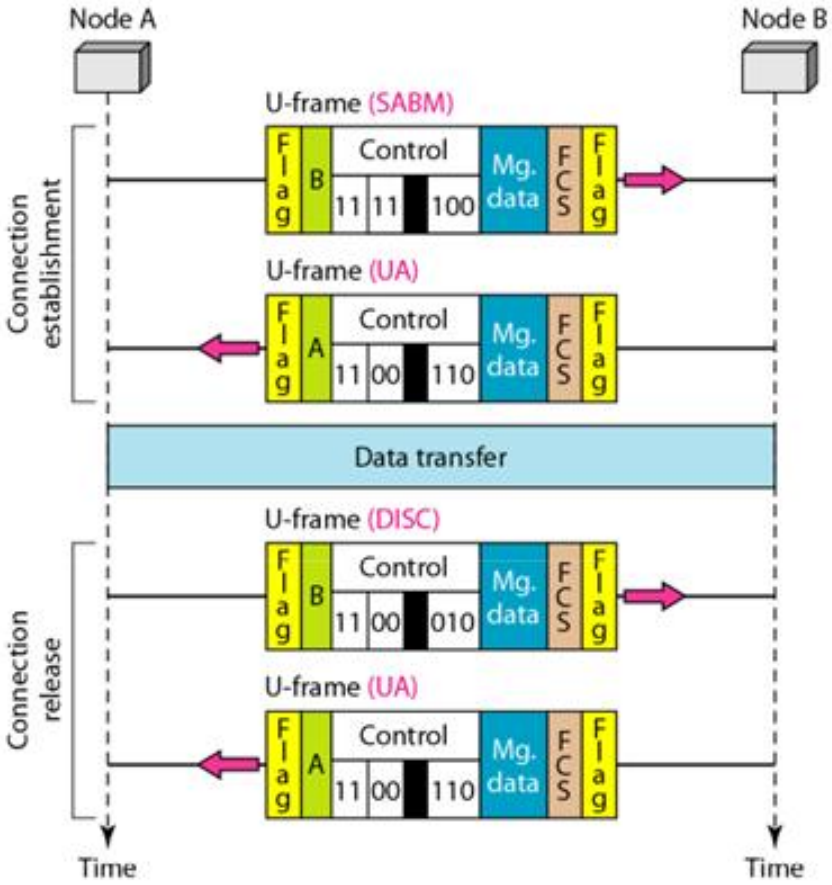
Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames. Some of the more common types are shown in below Table.

Uframe control command and response

Code	*Command*	*Response*	*Meaning*
00 001	SNRM		Set normal response mode
11 011	SNRME		Set normal response mode, extended
11 100	SABM	**DM**	Set asynchronous balanced mode or **disconnect mode**
11 110	SABME		Set asynchronous balanced mode, extended
00 000	UI	**UI**	Unnumbered information
00 110		**UA**	**Unnumbered acknowledgment**
00 010	DISC	**RD**	Disconnect or **request disconnect**
10 000	SIM	**RIM**	Set initialization mode or **request information mode**
00 100	UP		Unnumbered poll
11 001	RSET		Reset
11 101	XID	**XID**	Exchange ID
10 001	FRMR	**FRMR**	Frame reject

Example: *Connection/Disconnection*

Below Figure shows how V-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (VA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a VA (unnumbered acknowledgment).



POINT-TO-POINT PROTOCOL:

Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, one of the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP). Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point-to-point protocol at the data link layer. PPP is by far the most common.

PPP provides several services:

1. PPP defines the format of the frame to be exchanged between devices.
2. PPP defines how two devices can negotiate the establishment of the link and the exchange of data.
3. PPP defines how network layer data are encapsulated in the data link frame.
4. PPP defines how two devices can authenticate each other.
5. PPP provides multiple network layer services supporting a variety of network layer protocols.
6. PPP provides connections over multiple links.
7. PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

On the other hand, to keep PPP simple, several services are missing:

I. PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.

2. PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order.

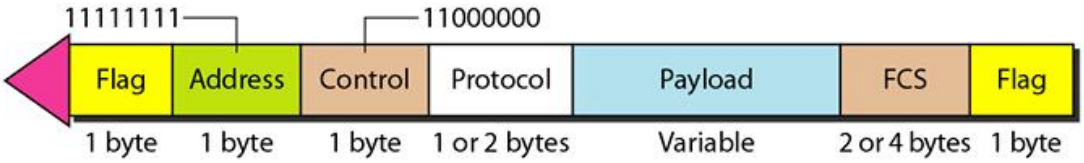
3. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

Framing:

PPP is a byte-oriented protocol. Framing is done according to the discussion of byte oriented protocols.

Frame Format

Below Figure shows the format of a PPP frame. The description of each field follows:



Flag. A PPP frame starts and ends with a I-byte flag with the bit pattern 01111110. Although this pattern is the same as that used in HDLC, there is a big difference. PPP is a byte-oriented protocol; HDLC is a bit-oriented protocol. The flag is treated as a byte, as we will explain later.

Address. The address field in this protocol is a constant value and set to 11111111(broadcast address). During negotiation (discussed later), the two parties may agree to omit this byte.

Control. This field is set to the constant value 11000000 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection. This means that this field is not needed at all, and again, the two parties can agree, during negotiation, to omit this byte.

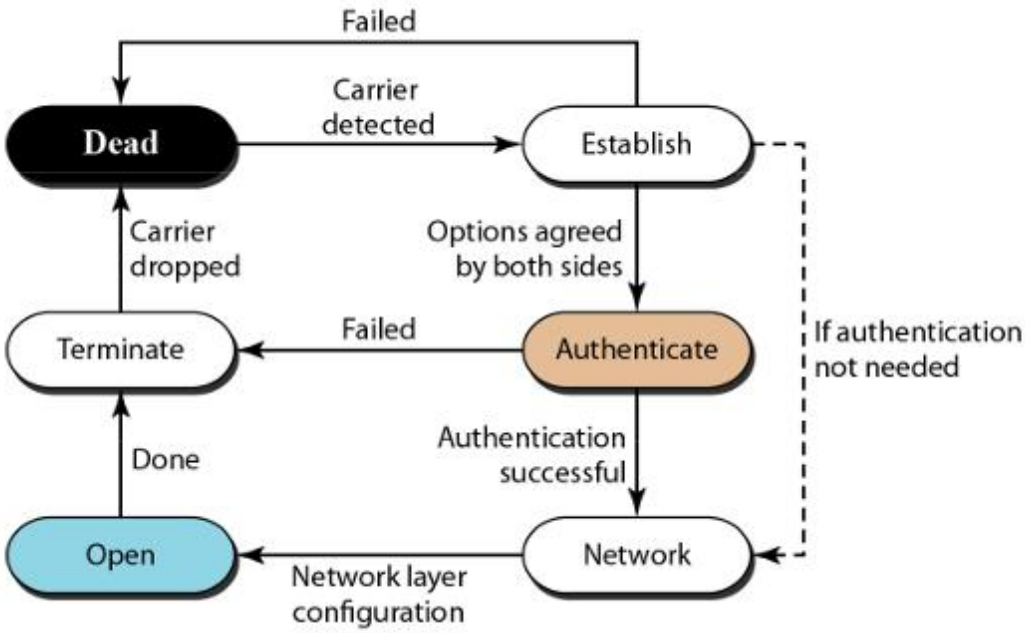
Protocol. The protocol field defines what is being carried in the data field: either user data or other information. We discuss this field in detail shortly. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.

Payload field. This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.

FCS. The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

Transition Phases

A PPP connection goes through phases which can be shown in a transition phase diagram.



Dead. In the dead phase the link is not being used. There is no active carrier (at the physical layer) and the line is quiet.

Establish. When one of the nodes starts the communication, the connection goes into this phase. In this phase, options are negotiated between the two parties. If the negotiation is successful, the system goes to the

authentication phase (if authentication is required) or directly to the networking phase. The link control protocol packets, discussed shortly, are used for this purpose. Several packets may be exchanged here.

Authenticate. The authentication phase is optional; the two nodes may decide, during the establishment phase, not to skip this phase. However, if they decide to proceed with authentication, they send several authentication packets, discussed later. If the result is successful, the connection goes to the networking phase; otherwise, it goes to the termination phase.

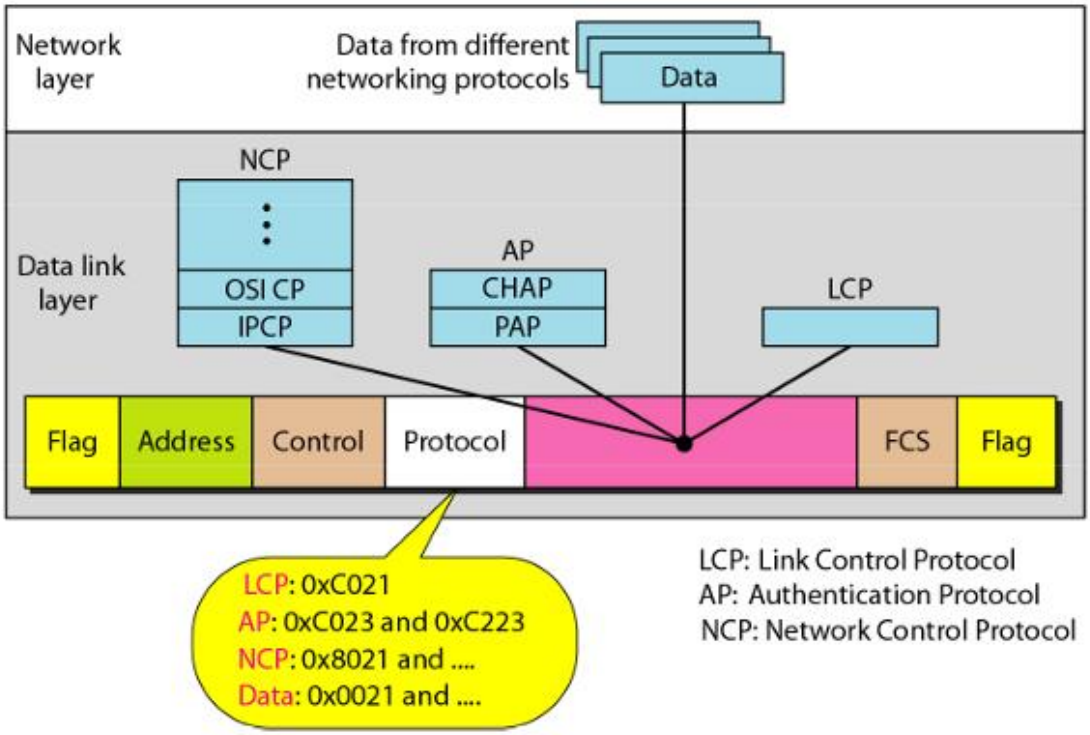
Network. In the network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at the network layer can be exchanged. The reason is that PPP supports multiple protocols at the network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive the data.

Open. In the open phase, data transfer takes place. When a connection reaches this phase, the exchange of data packets can be started. The connection remains in this phase until one of the endpoints wants to terminate the connection.

Terminate. In the termination phase the connection is terminated. Several packets are exchanged between the two ends for house cleaning and closing the link.

Multiplexing

Although PPP is a data link layer protocol, PPP uses another set of other protocols to establish the link, authenticate the parties involved, and carry the network layer data. Three sets of protocols are defined to make PPP powerful: the Link Control Protocol (LCP), two Authentication Protocols (APs), and several Network Control Protocols (NCPs). At any moment, a PPP packet can carry data from one of these protocols in its data field, as shown in below Figure. Note that there is one LCP, two APs, and several NCPs. Data may also come from several different network layers.

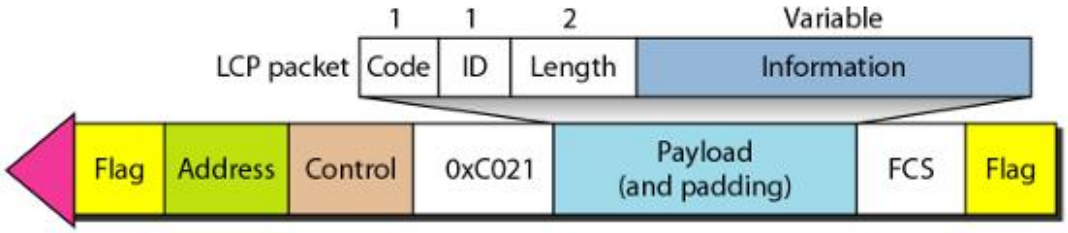


Link Control Protocol

The **Link Control Protocol** (LCP) is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established.

All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal.

The code field defines the type of LCP packet. There are 11 types of packets as shown in below Table.



Code	*Packet Type*	*Description*
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

There are three categories of packets. The first category, comprising the first four packet types, is used for link configuration during the establish phase. The second category, comprising packet types 5 and 6, is used for link termination during the termination phase. The last five packets are used for link monitoring and debugging.

The ID field holds a value that matches a request with a reply. One endpoint inserts a value in this field, which will be copied into the reply packet. The length field defines the length of the entire LCP packet. The information field contains information, such as options, needed for some LCP packets.

There are many options that can be negotiated between the two endpoints. Options are inserted in the information field of the configuration packets. In this case, the information field is divided into three fields: option type, option length, and option data. We list some of the most common options in below Table.

Option	*Default*
Maximum receive unit (payload field size)	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

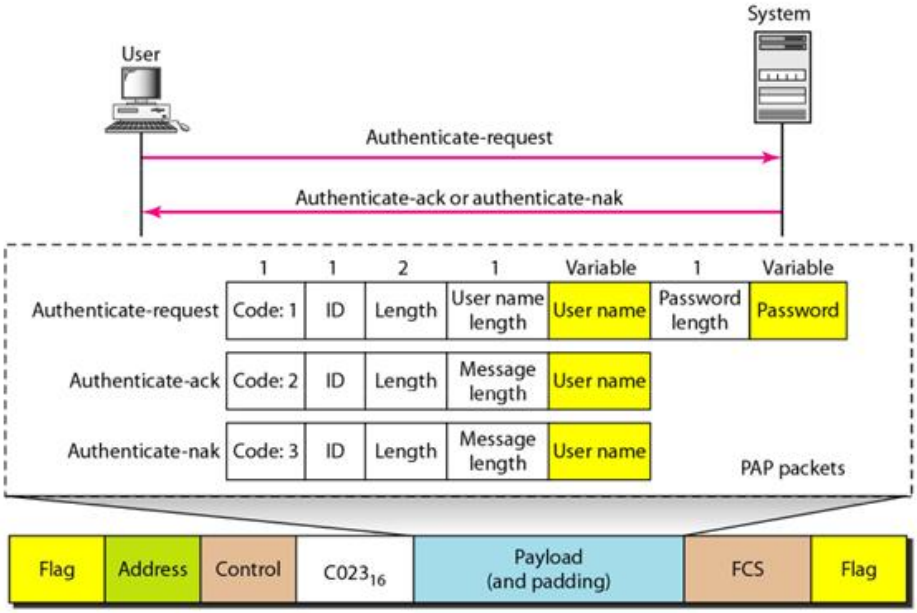
Authentication Protocols

Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. **Authentication** means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol.

PAP The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

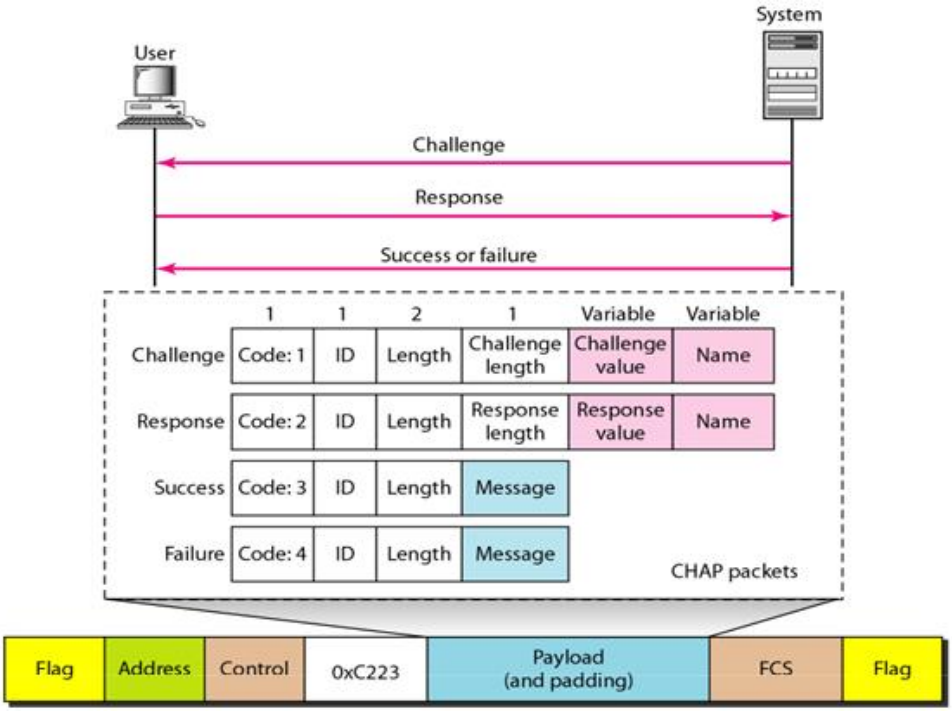
1. The user who wants to access a system sends an authentication identification (usually the user name) and a password.
2. The system checks the validity of the identification and password and either accepts or denies connection.

Below Figure shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is OxC023. The three PAP packets are authenticate-request, authenticate-ack, and authenticate-nak. The first packet is used by the user to send the user name and password. The second is used by the system to allow access. The third is used by the system to deny access.



CHAP The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way hand-shaking authentication protocol that provides greater security than PAP. **In** this method, the password is kept secret; it is never sent online.

1. The system sends the user a challenge packet containing a challenge value, usually a few bytes.
2. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.
3. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret. Below Figure shows the packets and how they are used.

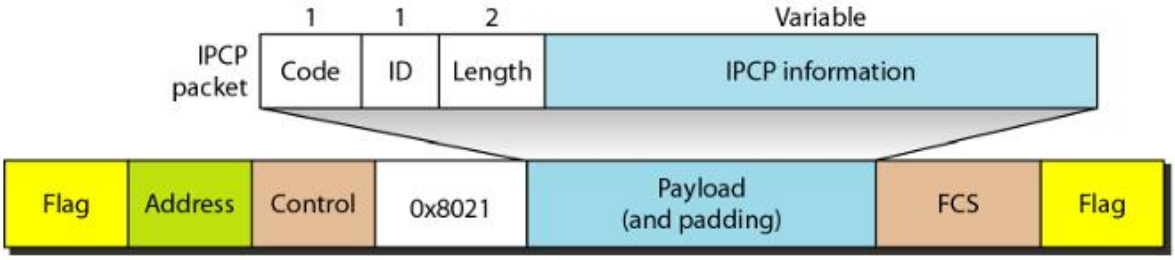


CHAP packets are encapsulated in the PPP frame with the protocol value C223 in hexadecimal. There are four CHAP packets: challenge, response, success, and failure. The first packet is used by the system to send the challenge value. The second is used by the user to return the result of the calculation. The third is used by the system to allow access to the system. The fourth is used by the system to deny access to the system.

Network Control Protocols

PPP is a multiple-network layer protocol. It can carry a network layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets. Xerox CP does the same for the Xerox protocol data packets, and so on.

IPCP One NCP protocol is the Internet Protocol Control Protocol (IPCP). This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in below Figure. Note that the value of the protocol field in hexadecimal is 8021.

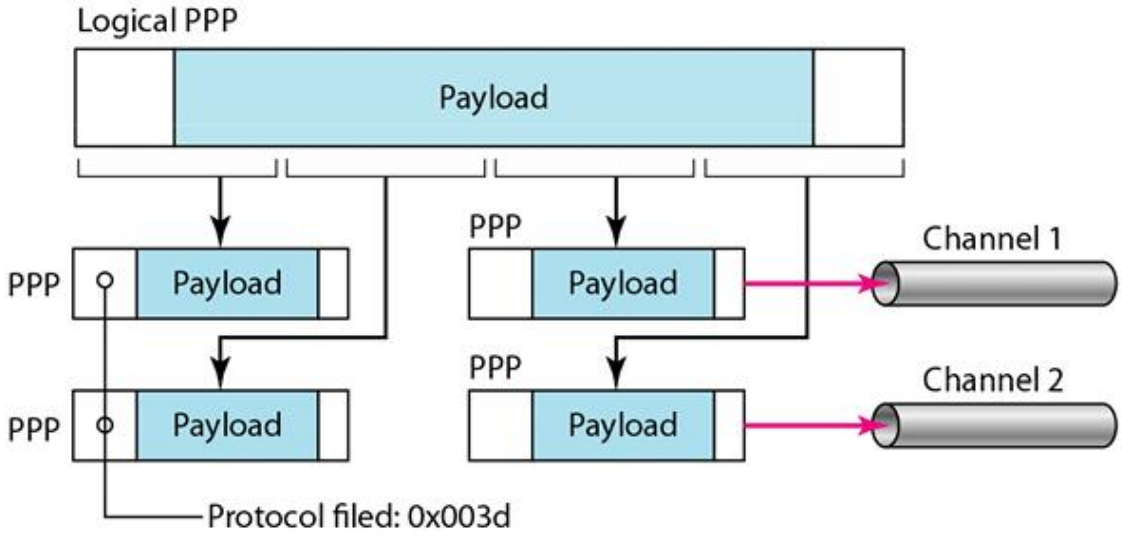


IPCP defines seven packets, distinguished by their code values, as shown in below Table

Code	*IPCP Packet*
OxO!	Configure-request
Ox02	Configure-ack
Ox03	Configure-nak
Ox04	Configure-reject
Ox05	Terminate-request
Ox06	Terminate-ack
Ox07	Code-reject

Multilink PPP

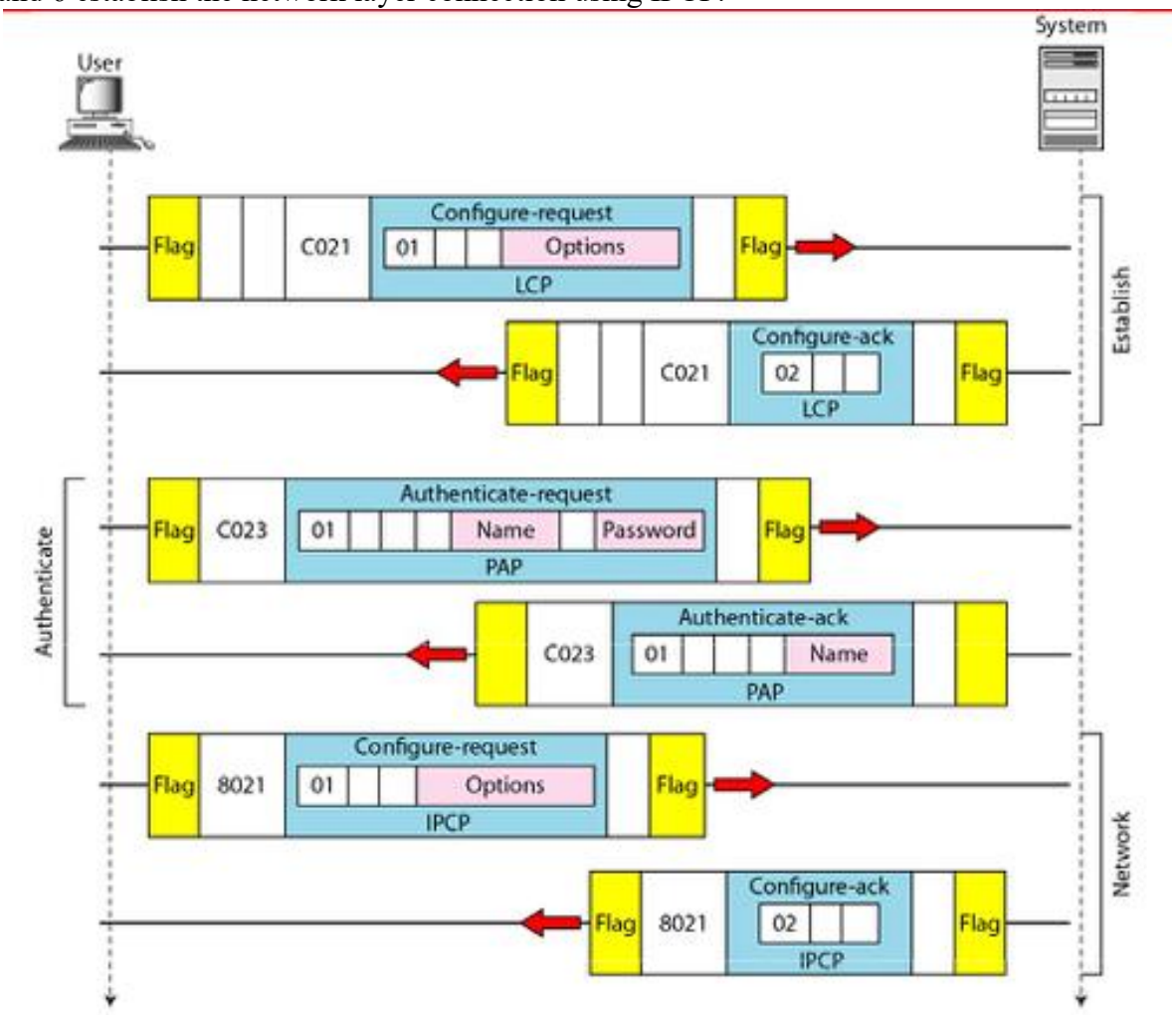
PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in below Figure. To show that the actual PPP frame is carrying a fragment of a logical PPP frame, the protocol field is set to Ox003d. This new development adds complexity. For example, a sequence number needs to be added to the actual PPP frame to show a fragment's position in the logical frame.

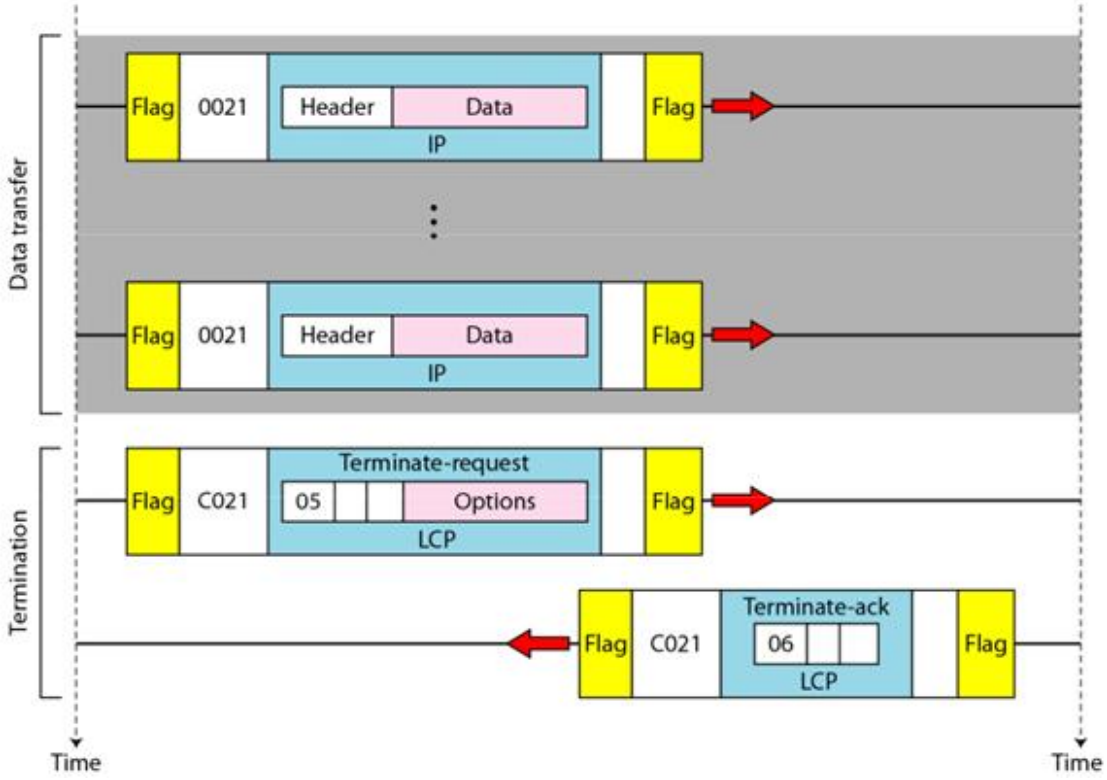


Example

Let us go through the phases followed by a network layer packet as it is transmitted through a PPP connection. Below Figure shows the steps. For simplicity, we assume unidirectional movement of data from the user site to the system site (such as sending an e-mail through an ISP).

The first two frames show link establishment. We have chosen two options (not shown in the figure): using PAP for authentication and suppressing the address control fields. Frames 3 and 4 are for authentication. Frames 5 and 6 establish the network layer connection using IPCP.




Data transfer
Flag
0021
Header
Data
IP
Flag
Flag
0021
Header
Data
IP
Flag
Termination
Flag
C021
Terminate-request
05
Options
LCP
Flag
Flag
C021
Terminate-ack
06
LCP
Flag
Time
Time

Unit - III

Routing - A packet is transmitted from source to destination in which path to reach the destination

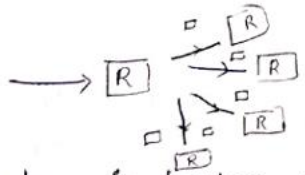
Routing Algorithms are classified in to 2 types

Non-adaptive Adaptive

Non-adaptive routing algorithms:-

- Also called static routing
- Routing process is designed in advance
- All the routing will be stored in routers when the routing is completed.
- It doesn't effect with change in N/w topology & traffic
- It is classified in 2 ways

1) Flooding - All incoming packages will be transmitted to all outing links except from the router it received.



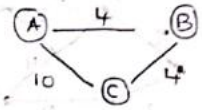
- Every router is having multiple copies of packets
- Random walk - incoming packets will be transmitted to the neighbour links randomly
- Random walk is best for alternative path

2) Adaptive Routing Algorithms:-

- Also called Dynamic routing algorithms
- Routing will change dynamically based on change in N/w topology & traffic.
- The main parameters are
  - Hopcount
  - Distance
  - Transmit time

  } based on these routing is decided

- classified in to 3 types
  1) centralized - Global routing algorithm. It Computes least cost path
  2) Isolation -
  3) Distributed



A → B → C is least cost path

- It is computed based on global info

Routing will be decided based on local information rather than Global

3) Distributed - It computes least cost path based an iterative & distributive manner.
- called as Decentralized algorithm

① Non-Adaptive Routing Alg
- Shortest - path
- Flooding
- Flow based

② Adaptive Routing Alg
- link state routing
- Distance vector (centralized)
  - Distributed routing

③ Hierarchical routing
④ Routing in mobile Hosts
5) Broadcast routing
6) Multicast routing

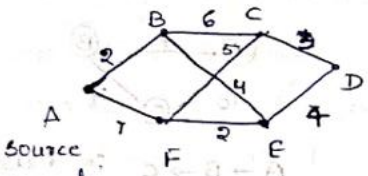
Non - Adaptive.

Shortest Path Routing Algorithms:-

The route in which packets travelling from source to destination in shortest route. With the help of function between nodes.

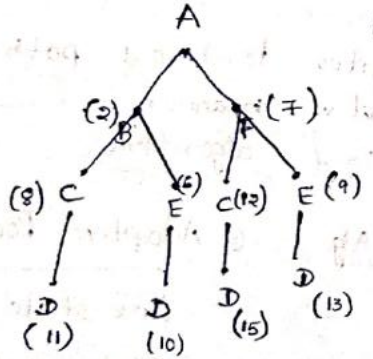
→ This function can be Host, Distance, Traffic & Bandwidth, Time

Ex:-



- Time as a function

Tree Structure



Paths

A B C D — 11

A B E D — 10

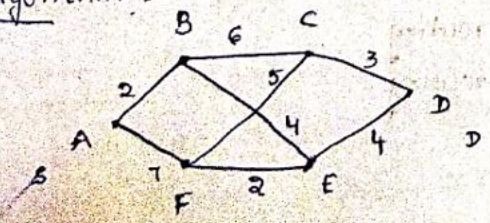
A F C D — 15

A F E D — 13

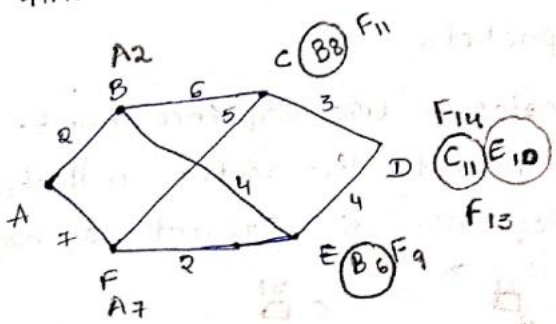
only forward directions

A B E D — 10 is the shortest path to reach the destination.

Dijkstraw Algorithm:-



Need to find incoming nodes



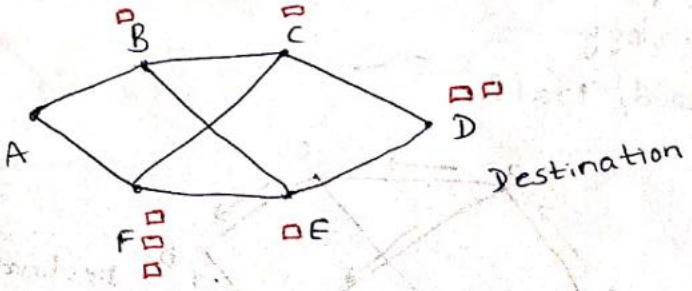
A → B → E → D

Total = 10 — shortest path.

Flooding:-

Broadcast the packet

- sends the packet to all outgoing links except to the link from which it was received.



A - B, F

B - ~~A~~, E

C - ~~B~~, D, F

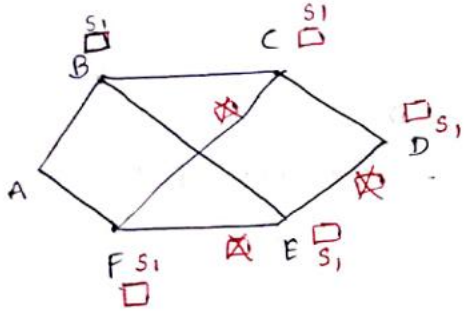
D - Destination

E - ~~B~~, F, D

Drawback :-

Duplicate packets

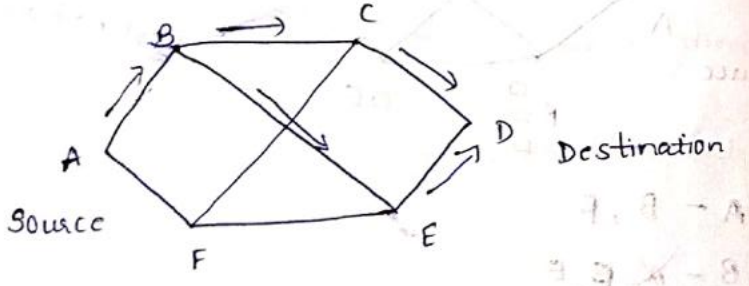
→ Avoid duplicates – use sequence number. (Before accepting a packet the router will check whether the same Seq no. is present or not)



Flow Based Routing :-

→ Routing is done based on two conditions

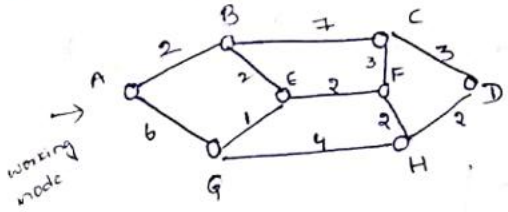
1) Topology
2) Load / Traffic.

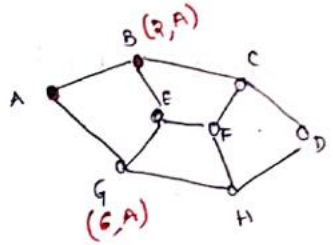


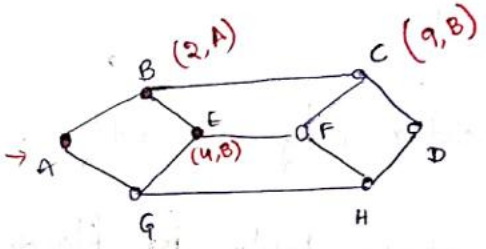
A - B - E - D shortpath routing - Having heavy Load / Traffic we have to select next shortest path

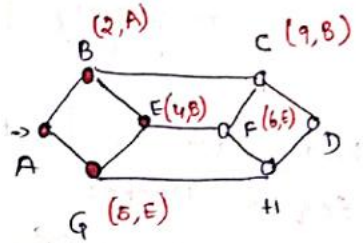
A - B - C - D next shortest path.

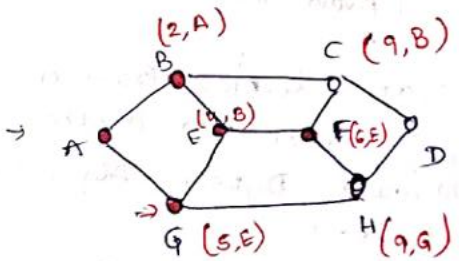
Example of Shortest Path routing Algorithm :-

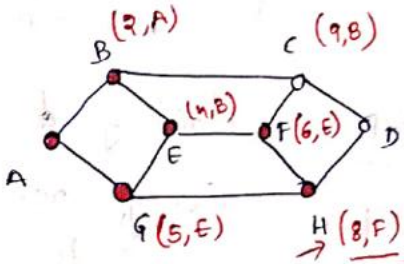












A - B - E - F - H - D =

O - Tentative nodes

● - Permanent nodes.

Adaptive Link State Routing Algorithm :-

It Consists of 2 phases
- flooding
- Routing path

Hierarchical Routing Algorithm:

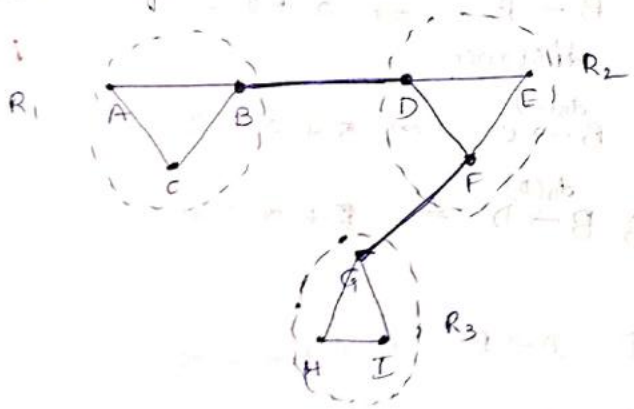
In HR, the routers are classified into d/f categories

collections of Routers - Regions
Regions - clusters
clusters - zones
zones - Groups.

A —msg→ B
Post

Address
↓
country
↓
city
↓
street
} Hierarchical

- Every router will know the information of all routers in same region



A — B, C
B — A, C
C — A, B

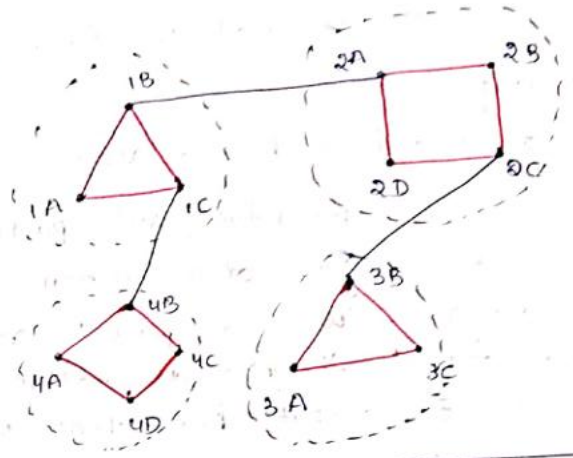
B, D are Gateway routers
F, G " " "

- Packets transferred from one region to another region through Gateway routers, to destination.

Advantages:

- It reduces complexity
- It reduces entries in routing table
- Routing efficiency is increased

Ex:-



Source 1A

Destination	Next hop	No. of hops
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	4
2D	1B	3
3A	1B	6
3B	1B	5
3C	1B	6
4A	1C	4
4B	1C	2
4C	1C	3
4D	1C	4

14 entries

Source 1A

Destination	Nexthop	No of hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1B	4
4	1C	3

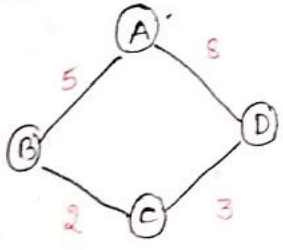
Hierarchical routing table with 6 entries (reduced entries)

routing table — Destination hop
Distance
next hop

- This info. is maintained in each & every router
- " " is shared with all neighbouring routers
- The routing info. will be updated periodically.
- This is updated until no any new info. to exchange
- It follows Bellman-Ford equation inorder to calculate least-cost path.

$$d_x(y) = \min_v \left(\text{cost}(x,v) + d_v(y) \right)$$

x - source
y - destination
v - intermediate router.



A

De	Di	NH
A	0	A
B	5	B
C	∞	—
D	8	D

B

De	Di	NH
A	5	A
B	0	B
C	2	B
D	∞	—

C

De	Di	NH
A	∞	—
B	2	B
C	0	C
D	3	D

D

De	Di	NH
A	8	A
B	∞	—
C	3	C
D	0	D

second iteration is shared among neighbouring routers.

updated routing table A < B, D

A

De	Di	NH
A	0	A
B	5	B
C	7	B
D	∞	—

B

Di
5
0
2
∞

A

De	Di	NH
A	0	A
B	∞	—
C	11	D
D	8	D

D

8
∞
3
0

Here we need to consider min. cost & min. intermediate nodes.

A→B: A → B, $d_B(B)$: B → B ⇒ 5 + 0 = 5 (cost : distance)

A→C: A → B, $d_B(C)$: B → C ⇒ 5 + 2 = 7

A→D: A → B, $d_B(D)$: B → D ⇒ 5 + ∞ = ∞

} A → B

A→B: A → D, D → B ⇒ 8 + ∞ − ∞

A→C: A → D, D → C → 8 + 3 − 11

A→D: A → D, D → D ⇒ 8 + 0 − 8

} A → D

combine both tables & find least cost path

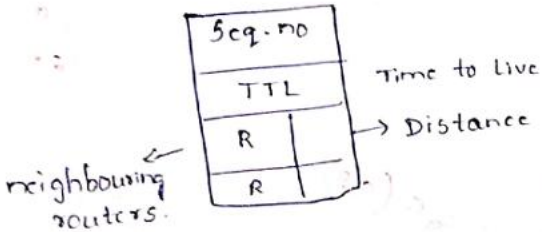
A

De	Di	NH
A	0	A
B	5	B
C	7	B
D	8	D

updated

Step 1:- finding distance for neighbouring routers

Step 2:- Find link state table for every router



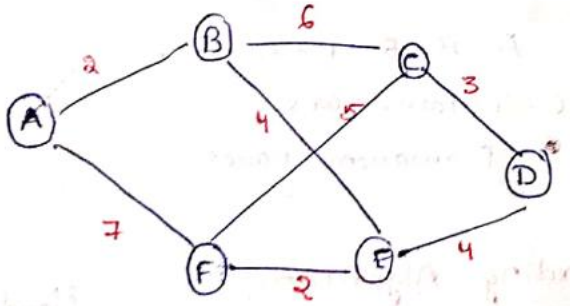
Here we need to consider only recently visited Seq.no Packet.

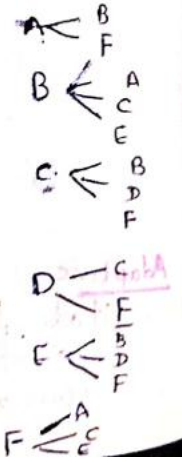
Time to Live - The link state tables are flooded among all routers and the flooded value goes on decreasing from one router to another router.

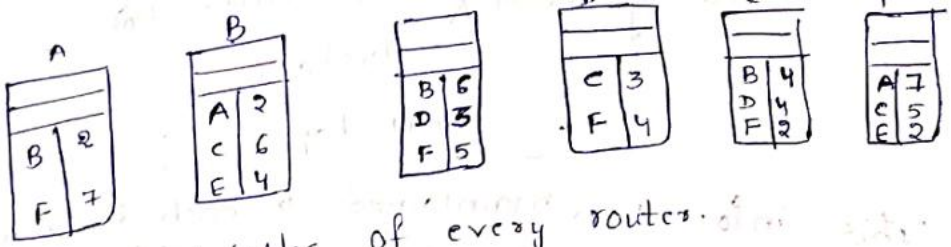
TTL > 0 (Avoid infinite flooded).

Step 3 - Flooding of link state tables from one router to another.

Step 4 - Every router follows Dijkstra Algorithm for shortest path.







Link state tables of every routers.

→ These are flooded among routers

A (0) - B (2), F (7)

AB - C (8), E (6)

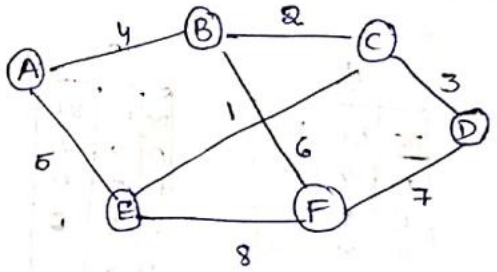
ABE - D (10), F (8)

ABEF - C (13)

ABEFC - D (16)

ABEFCD - shortest path

Ex:-



Distance Vector Routing Algorithm:-

- No of routers are present to transfer pckt from router to router
- It is dynamic routing Algorithm
- every router will maintain routing table.

Broad Cast Routing :-

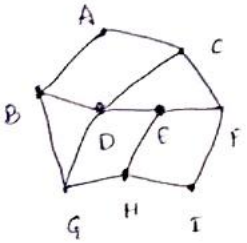
Sending a packet to all destinations simultaneously is called Broadcasting.

- several methods of Broadcast Routing are :

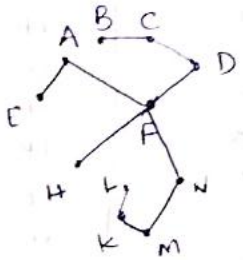
a) Multi destination routing : in which each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be required

b) Flooding : when implemented with a sequence number per source, flooding uses links efficiently with a decision rule at routers that is relatively simple. It is point-to-point communication.

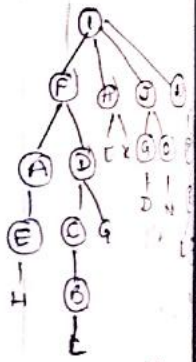
c) Reverse path forwarding : Sink tree



(a)



(b)



(c)

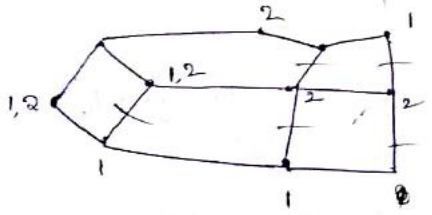
Spanning tree (sink tree) is a subset of n/w that includes all routers but contains no loops. If each router knows which of it's lines belong to spanning tree, it can copy an incoming broadcast packet onto all spanning tree lines except the one it arrived on. This method makes excellent use of bandwidth, generating the absolute min. no. of packets necessary to do the job

Multicast Routing

Sending a message to a group (large) is called multicasting

→ multicasting requires group management need to create or destroy groups & to allow processes to join & leave groups

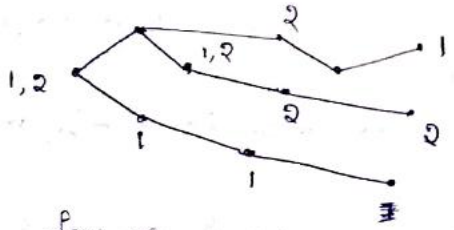
→ To do multicast routing each router computes a spanning tree covering all other routers.



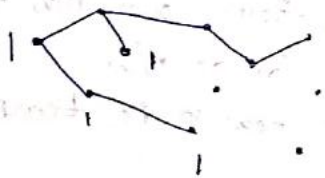
1, 2 ← 2 Groups
2 – 3 .
1 – 3 Groups

↓

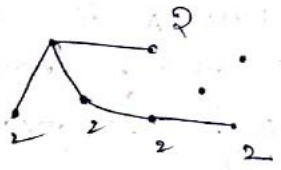
Spanning tree



for group 1



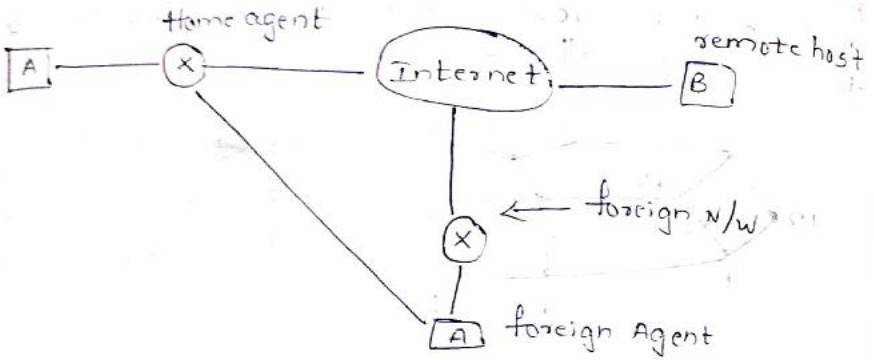
for group 2



Routing for mobile host:-

Routing from source to destination. If it is a device, it sets ip addresses

1) Agent discovery
2) Registration
3) Data transfer



Changing the IP, when mobile device change its one point to another point.

→ machine 'A' is present in internet, here router in Home n/w called Home agent.

→ when we start a device into home n/w, machine will undergo 3 phases

Agent Discovery – When machine 'A' requests for home agent to assign permanent Ip, then home agent assign home ip address to 'A'.

In future 'A' is shifted from own n/w to another n/w is actually known as foreign n/w.

Here 'A' is assigned with new ip i.e, temporary ip foriegn n/w

- Permanent ip is valid when it is in Home n/w only.

Registration:- The 'A' will request to foreign agent to assign something known as care of Ip address.

- Foriegn agent sends the request to Home n/w whether the device is valid or not. If it says 'yes' then home agent sends +ve acknowledgement.
- Then temperary IP is assigned known as 'care of Ip address'.

Data transfer:- when 'B' is communicating with 'A' in home n/w but doesn't know that 'A' is in foreign n/w.

Congestion Control ~~Algorithms~~:-

Routing was main functionality of n/w layer & second functionality is Congestion control

Approaches to congestion control

- N/W Provisioning.
- Traffic - Aware routing
- Admission control
- Traffic Throttling
- Load shedding

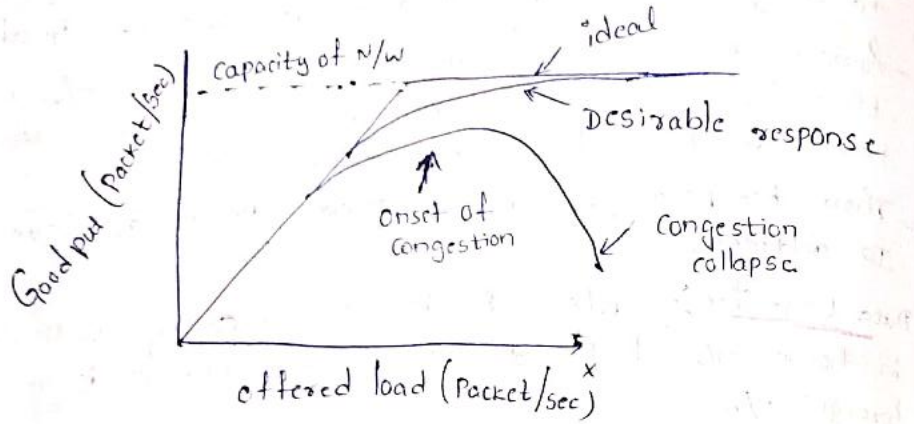
Congestion - when too many packets are present in the n/w, performance degrades.

- Both N/W & Transport layer has the responsibility to control Congestion

N/W service ~~Servic~~ Connectionless - Congestion is not controlled

Connection oriented - Congestion is controlled

It is left to Transport



→ when too much traffic is offered, Congestion sets in & performance degrades sharply.

Congestion occured due to:

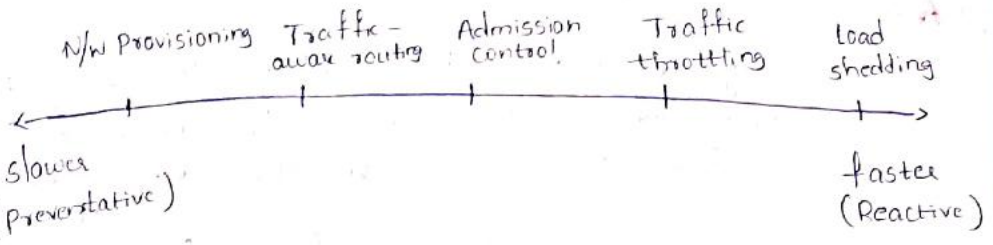
- Insufficient memory in routers Ex:- queue is full
- Slow Processors Ex:- CPU, routers are slow at processing
- Low Bandwidth lines

Congestion Control Vs Flow Control

Congestion control - making sure that the entire subnet is able to carry the offered traffic
↓
Routers & hosts will face the traffic

Flow control - It is b/w source & corresponded receiver the path b/w source & Destination

- Approaches to congestion Control is either increase resources or decrease the load



Timescales of approaches to Congestion Control

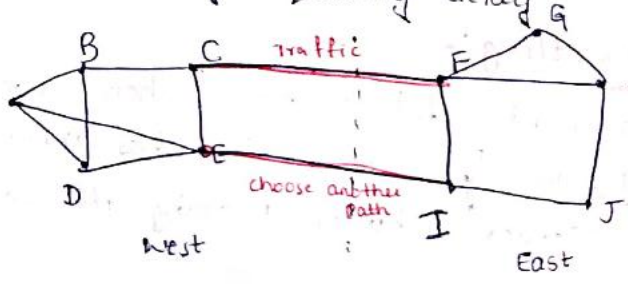
1) N/w Provisioning -

To build a n/w that is well matched to traffic that it carries.

- Spare routers that are normally used only as backups
- bandwidth on open market need to be purchased
- links and routers that are regularly heavily utilized are upgraded

2) Traffic aware Routing -

- routes can be tailored to traffic patterns
  - that change during the day as n/w users wake & sleep in d/f time zones.
  - splitting traffic across multiple paths.
- set the link weight to be a function of link bandwidth & propagation delay plus the measured load or average queuing delay



a) Random Early Discard (RED):

It is proactive approach
→ Congestion prevention.
→ Here in this approach in which routers discards one or more pckts before the buffer becomes completely full.

Inter Networking:

It is the process or technique of connecting d/f n/w by using intermediate devices such as routers, switches, gateways etc

how n/w's differ

i) Services offered - It can be connection oriented or connectless services.
ii) Protocol used - Some possible protocols are IP, CLNP, DEC, Apple talk etc
Digital Equipment Corporation
iii) Addressing - It can be flat (802), or hierarchical (IP)
iv) Multicasting - It can be present or absent
v) Packet Size - It depends on the n/w
vi) Error handling - we have reliable ordered, or unordered delivery
vii) flow Control - we can get sliding window, rate control etc
viii) Congestion Control - D/f algorithms (Explicit Cong. Notification), choke packets are available.

How n/w's Can be Connected:

Based on layers d/f Components are used

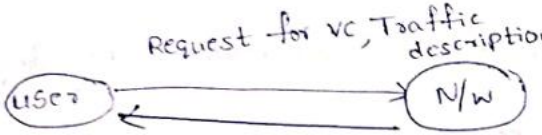
i) Repeaters - It works at physical layer. These are used for amplifying & strengthening weak signals. Can travel over long distance
ii) Bridge - It works at data link layer. It is an intelligent device that are used to interconnect local area n/w's.
- The main fn. of a bridge is when a frame is received, it checks the destination mac address of frame. if it finds the particular destination on other side of lan, then only frame is forwarded
- Hence bridge takes decision.
iii) Routers - The routers works in n/w layer so the data can be transmitted across multiple n/w
iv) Transport Gateway - This is used in transport layer for connecting 2 n/w's in that layer
v) Application Gateway - These are used in application layer for translating message semantic

Types of internetworking:-

i) Concatenated virtual circuits
ii) Connectionless internetworking

3) Admission Control

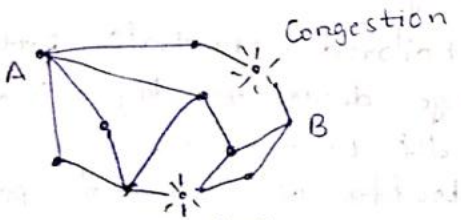
- Widely used in virtual-circuit n/w. (Sender request for virtual circuit N/w. If it is established packets follow the virtual circuit N/w to transfer)



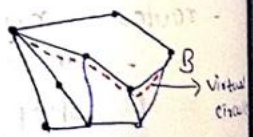
Request granted if the traffic can be accomodated without congestion, else denied.

- It characterize traffic
- reserve enough capacity along the paths of each of its virtual circuits.

Combining admission control with Traffic aware routing



Traffic away routing



Admission Control

4) Traffic Throttling:-

Routers must determine when congestion is approaching, ideally before it has arrived & warn the sender who is making this congestion.

→ Each router can continuously monitor the resource it is using.

resources:

1) utilization of output links
2) buffering of queued packets inside the router
3) no. of packets that are lost due to insufficient buffering

Estimate of the queuing delay (EWMA (exponentially weighted Moving Average))

$$d_{new} = \alpha d_{old} + (1-\alpha)s$$

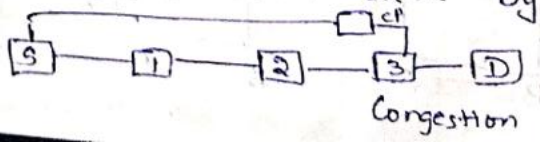
(d_{new}: newest feature; d_{old}: previous history)

d = Estimate of the queuing delay
s = a sample of instantaneous queue length
α = determines how fast the router forgets recent history
α value should be b/w 0 & 1.

a) Choke Packet: A more direct way of telling the source to slow down.

→ A choke packet is a control packet generated at a congested node & transmitted to restrict traffic flow.

→ The source on receiving the choke packet must reduce its transmission rate by a certain percentage

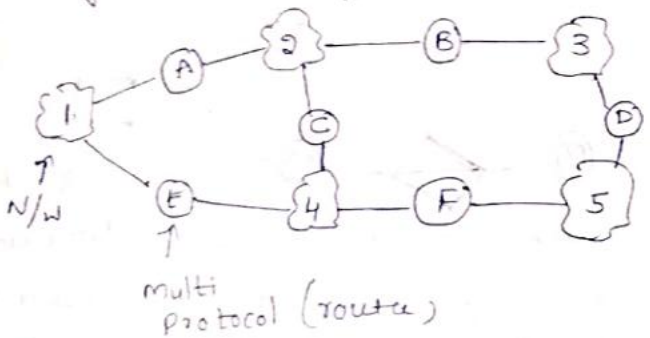


Tunneling

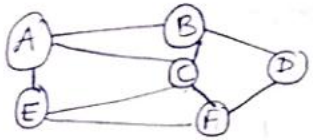
Tunneling is a protocol that allows for the secure movement of data from one n/w to another
→ Data pckts follow point to point tunneling (PPP Protocol)
→ Tunneling involves allowing private n/w Communications to be sent across a public n/w such as internet.

Internetwork Routing

Routing through an internetwork is similar to routing within a single subnet, but with some complications



5 N/W connected to 6 routers



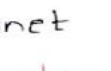
→ Converting into graph we can apply routing

Fragmentation

The n/w designers are not free to choose any size of pckt
→ The maximum pckt size varies n/w to n/w & the factor that decides, max. pckt size
i) hardware (Eg. size of ethernet frame)
ii) Operating system (all buffers are 512 bytes)
iii) Protocols (No. of bits in the pckt length field)
iv) Based on international standards
v) efforts to reduce retransmission
vi) Desire to prevent a pckt from occupying the channel too long"
→ All these ~~function~~ factors put a limit on maximum packet size
→ The max payload size is 48 bytes for ATM cell to 65 to 515 bytes for an IP packet

fragmentation — In this technique the gateways break up large packets into smaller ones also called as fragments
→ Then each fragment is sent as a seperate internet packet.

Recombination of fragments — (a bit difficult)



The recombination of fragment can be done by using two strategies.

i) Transparent fragmentation

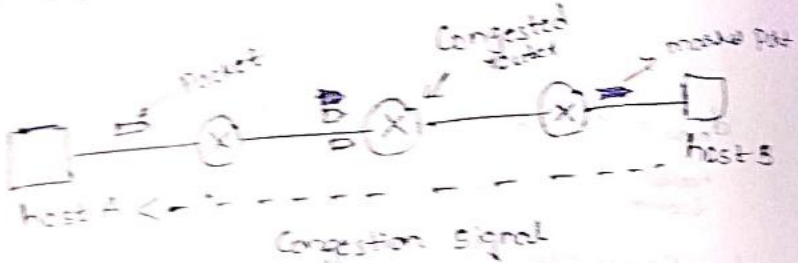
In this strategy, the fragmentation caused by a small packet n/w is made transparent to any subsequent n/w through which pckts will pass.
→ when a large pck arrives gateway it breaks pckt in to fragment.

hop-by-hop choke packet :-

Over long distance or at high speed choke packets are not very effective

c) Explicit Congestion Notification :-

- Using warning bits in packet header
- destination notes that there is congestion & inform the sender in its reply.



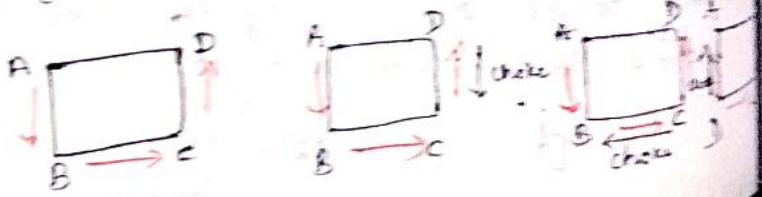
d) Hop by hop back pressure :-

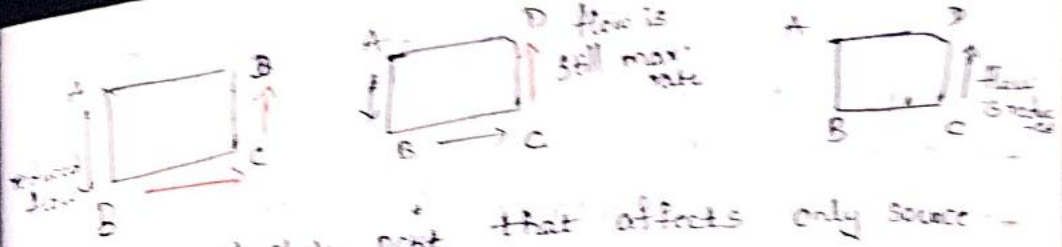
At high speeds or over long distances, sending a choke ~~speed~~ packet to source hosts doesn't work well because reaction is slow

Alternative approach:

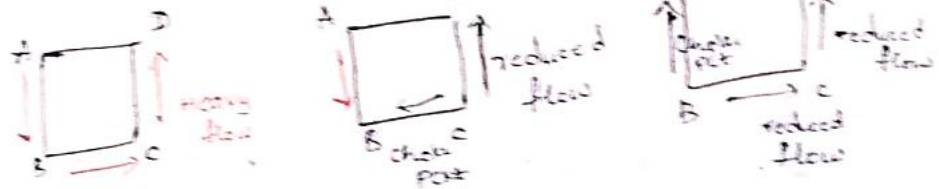
To have choke packet take effect at every hop it passes through.

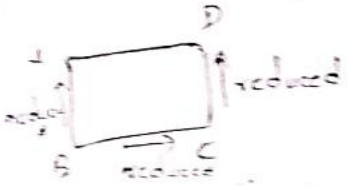
Direct choke packets hop-by-hop Back pressure





a) Choke point that affects only source :-





b) A choke pckt that affects each hop it passes through

e) Load shedding :-

It is one of the most useful one in Congestion occurance

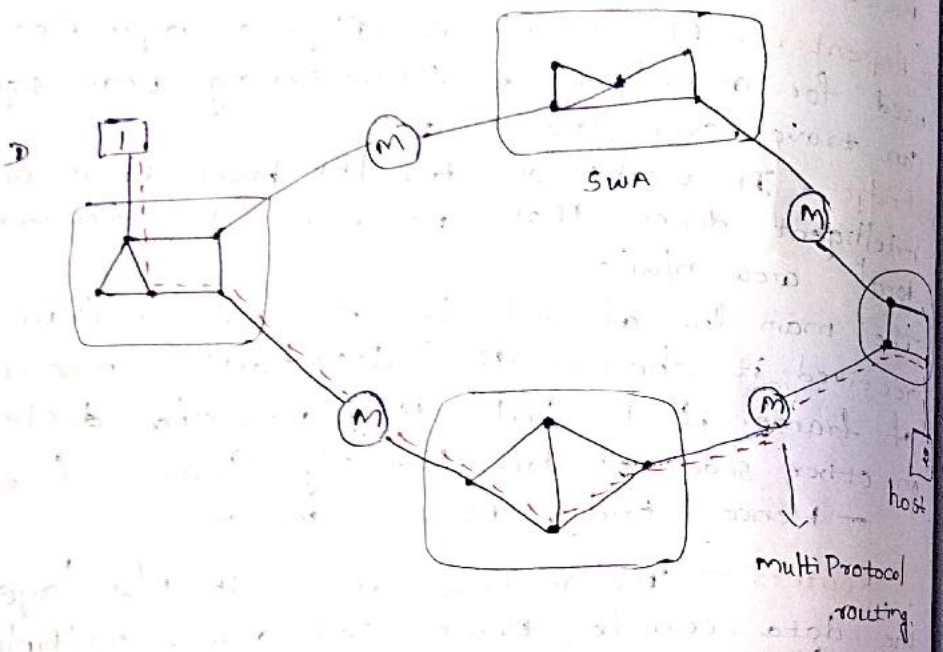
→ when buffer becomes full, routers simply discard packets

→ which pckts is choosen to be the victim depends on the application & on the error strategy used in data link layer.

Ex:- ① For a file transfer, we cannot discard one pckt since that will cause a gap in received data

② For real time voice or video chat app it is better to throw away old data & keep new

i) Concatenated virtual circuit
(or)
Connection oriented circuit:



In the concatenated virtual circuit model shown in above fig, a connection to a host in a distant n/w is set up a way similar to the way connections are normally established.

→ The subnet sees that the destination is remote & builds a virtual circuit to router nearest to destination n/w.

→ Then it constructs a virtual circuit from that router to an extend gateway (multiprotocol router)

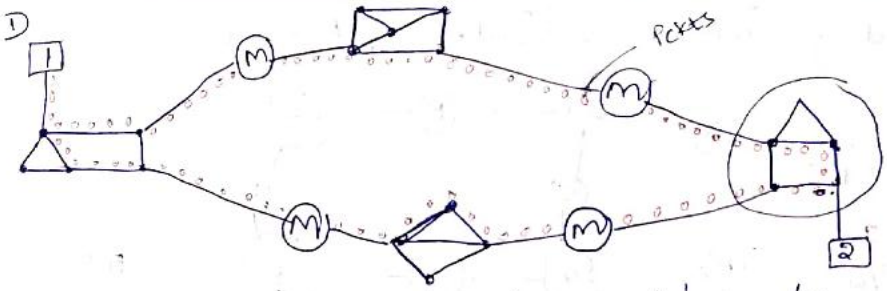
→ This gateway record, the extend of the VC in it's table & proceed to build another VC to router in next subnet.

→ The Process continues until destination host has reached.

Adv: Buffers can be reserved in advance
seq. can be guaranted
short headers can be used.

Dis:- Table space is needed
→ No other route possible, if congestion occurs.

Connectionless internetworking / Datagram



The alternative internetwork model is datagram mod

→ In this model, the only service the n/w layer offers to the transport layer is the ability to inject datagram into the subnet & hope for the best.

→ All pckts may or may not go in same path

→ In diagram we shown paths 1 & 2

→ Routing decision is made for each packet (depending on traffic)

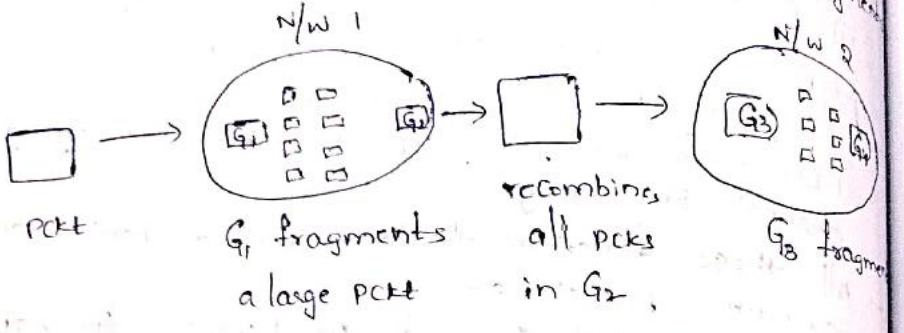
→ Ordering can also change

Adv :- Higher band width than connected

Disad :- each pckt. take seperate path not in particular order is maintained.

→ Each fragment is then addressing to the same exit gateway

→ The exit gateway recombines all these fragments



G_1 fragments a large pckt

G_3 fragments

Disad/Drawbacks

→ All pckts should exit via the same gateway

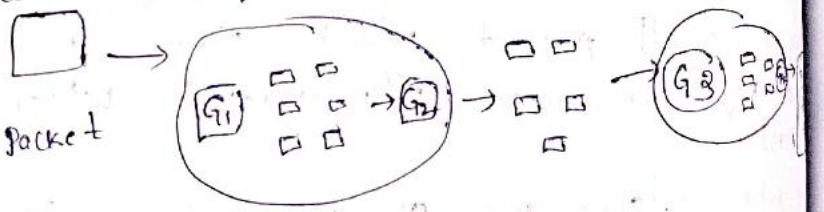
→ High overhead required to repeat & reass the packet

Non Transparent fragmentation:

In this strategy, the fragmentation packets are not reassembled at all.

Instead each fragment is treated as a sepa original pckt.

→ All these pckts are passed through exit gatew recombination is carried out at the destination



Disad/draw backs -

→ Every host must be capable of reassembling fragments.

→ Total over head increases since each fragment has to have a header

→ pckt is fragmented hence they should be numbered in such a way that original data stream can be reconstructed

Adv:-

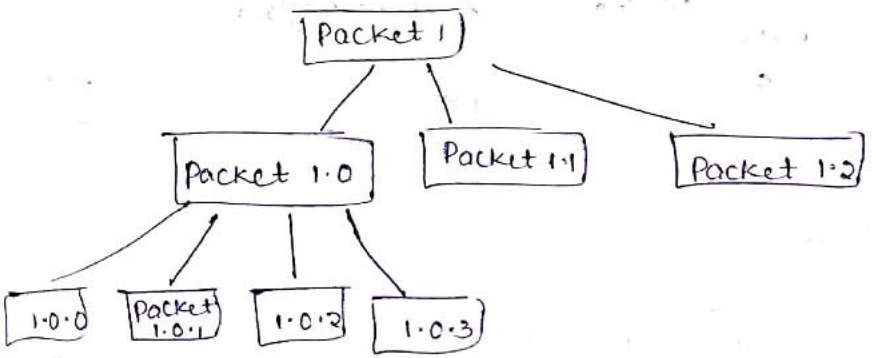
Better performance than transparent

Few numbering individual fragment pckts:

i) using tree structure:

If pckt is divided into fragments they are represented as 1·0, 1·1, 1·2, ... if then are fragmented again into. smalled fragments numbe

1·0·0, 1·0·1, 1·0·2



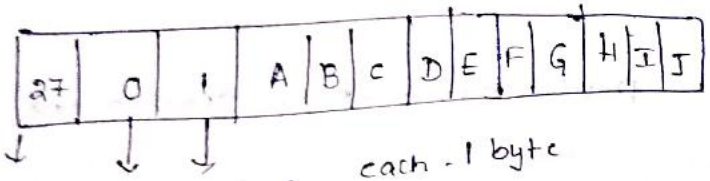
If any fragment lost in middle, retransmission is done. If pckt which was retransmitted goes via another n/w which causes prblms in reassembl

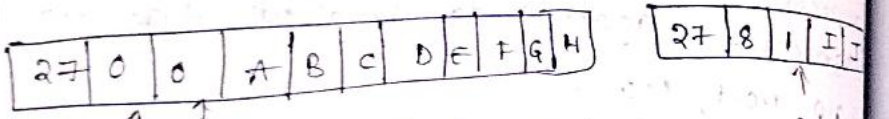
ii) Defining elementary fragment size -

using this type of addressing each pckt will be fragmented equally except last one.

→ Each Pckt may contain several fragments

→ Each pckt contains info. about original pckt.





The N/W :

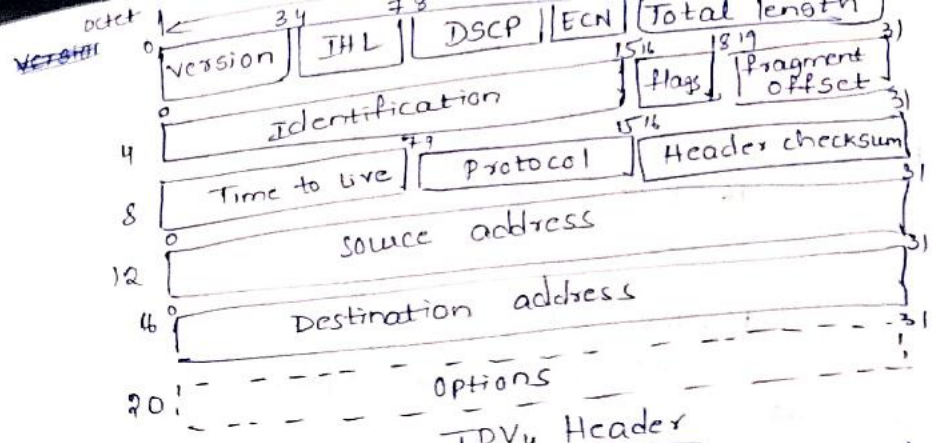
IPV4 Header / IP Protocol

Two d/f protocols IPV4 & IPV6

↓

most commonly / previously using

Internetworking Protocol Version 4



Version - These 4 bits specify the version of IP which we are using

IPV4 or IPV6

↓ ↓

0100 0110

IHL/HLEN - (4 bit field)

The IHL field specifies the actual length of the header in multiples of 4 bytes / 32 bits

→ we know header size/length can vary b/w 20-60 bytes

IHL is valid only from 5 to 15

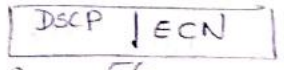
5 indicates 5×4 = 20 bytes

15 " 15×4 = 60 bytes

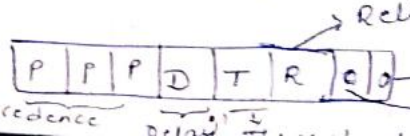
Type of services field / DSCP (8 bit)

Differentiated Service Code point

DSCP → two parts



→ Explicit Congestion Notification



Total length (16 bit)

The total length field defines the total length of initial datagram including the header & Payload

Ports

16 bits ⇒ 2^{16} = 65535

header size 20 - 60 bytes

Payload 0 - 65515 bytes

Identification (16 bit field)

It specifies the identity of a fragment i.e, to which datagram it belongs to.

Flags (3 bit field)

Reserved	Don't fragment	more fragment
Resave	DF	MF

1 - Reserved

0 - Not reserved

DF :- 0 } fragment

0 } fragment

1 } No fragment

MF

0 } if last fragment

1 } else MF = 1

Fragment Offset : (13 bit field)

It specifies the location of fragment in datagram (how much data before the fragment)

Time to Live (8 bits) - (0 - 255) bits

which refers to a counter used for limit the lifetime of a pckt.

(0 - 255 sec)/hop : is the lifetime of a pckt

Protocol - (8 bit)

It specifies to which transport layer protocol (TCP or UDP) datagram is to be given.

Header checksum - (16 bit)

which checks the header part, so to detect the error occurred during the transmission of pckt

Source & Destination address (each 32 bit)

Source address specifies port no. of host (sender)

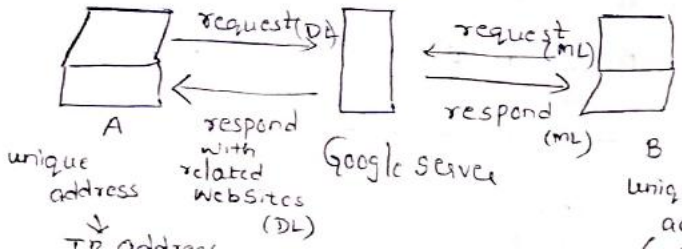
Destination " " " " (receiver)

Options field (32 bit)

Options field is used in selected datagram to carry additional info. relating to security, source routing, route routing, route recording, stream identification & time_stamp etc.

IP address :-

Address of identity for the device. for Communicate with other devices over n/w we require ip addresses.



IP address
↓
internal Protocol
↓
Set of rules to be followed.

↓ 4 formats (parts)

— . — . — . — } IPV4

↓ each part is having

0 to 255 | 0 to 255 | 0 to 255 | 0 to 255

we should represent with 8 bits (11111111) (00000000)
255 0

because System will recognize only binary format

→ length of IP address in IPV4 is 32 bits ∴ 2^{32} ip addresses are generated (2^{32} devices)

IPV6 - 8 parts → — . — . — . — . — . — . — . —
16 16 16 16 16 16 16 16

↓ (hexadecimal) (0-9, A-F) → 16 bits

Total length of IP in IPV6 is 128 bits → 2^{128}

classes of IP address :-

IP addresses are divided into d/f types

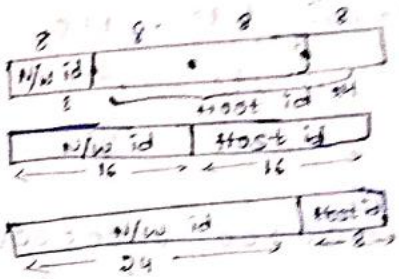
class A, B, C, D, E

— . — . — . —
↓ ↓ ↓ ↓
(0-255) 8 bits

IP address
↓ ↓
n/w id | host id

n/w1 | n/w2 | n/w3

class A - 32 bits
B - "
C - "
D - "
E - "



Multicasting — D
Research & Experiments — E

class A

Leading bit — 0 (1st bit)

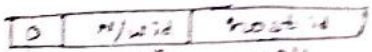
(7 bits for hosts)

00000000 . 00000000 . 00000000 . 00000000
to
01111111 . 00000000 . 00000000 . 00000000

n/w id

range → 0.0.0.0 to 127.0.0.0

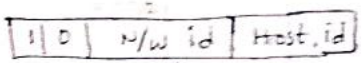
(0 to 127) — class A



Subnets ⇒ 2^7
host ⇒ 2^{24}

class-B

leading bits are 10 (2 bits)

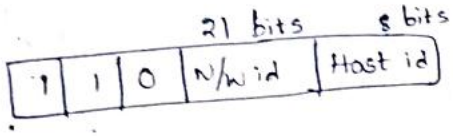


Subnets Host

range
128 to 191

128 10000000 . 00000000 . 00000000 . 00000000
to
191 10111111 . 00000000 . 00000000 . 00000000

class C



leading bits - 110

Subnets - 2^{21}

hosts - 2^8

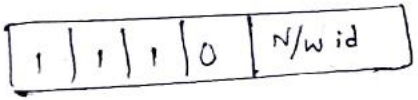
11000000 . 00000000 . 00000000 . 00000000

to

110 11111 . 00000000 . 00000000 . 00000000

Range ⇒ 192 to 223

class D -



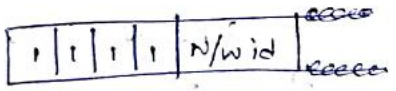
leading bits - 1110

11100000 . 00000000 . 00000000 . 00000000

1110 1111 . 00000000 . 00000000 . 00000000

Range - 224 to 239

class E -



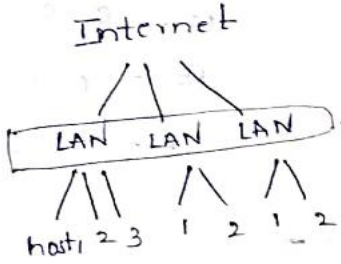
11110000 . 00000000 . 00000000 . 00000000

to

11111111 . 00000000 . 00000000 . 00000000

class	leading bits	Range of IP address
A	0	0 to 127 ⇒ 7 - N/w id, 24 - host id
B	10	128 to 191 ⇒ 14 - N/w id, 16 - host id
C	110	192 to 223 ⇒ 21 - N/w id, 8 - host id
D	1110	224 to 239 → Multicast
E	1111	240 to 255 → Research & Experiments.

Subnet Mask



for subnet
Making N/w bits-
host bits-

class A - 8 bits - 24 bits — 11111111 . 00000000 . 00000000 . 00000000
N/w id | host id
255 . 0 . 0 . 0

class B - 16 bits - N/w id — 11111111 . 11111111 . 00000000 . 00000000
16 bits - host id
255 . 255 . 0 . 0

class C - 24 bits - N/w id — 11111111 . 11111111 . 11111111 . 00000000
8 bits - host id
255 . 255 . 255 . 0

0 → represents N/w
255 → Broadcast purpose in each & every class.

0 to 255 ⇒ 256

leading bits - 110

Subnets - 2^{21}

hosts - 2^8.

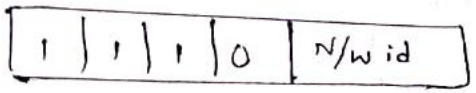
11000000.00000000.00000000.00000000

to

11011111.00000000.00000000.00000000

Range ⇒ 192 to 223

class D.



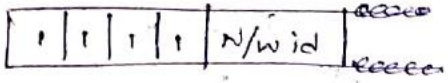
leading bits - 1110

11100000.00000000.00000000.00000000

11101111.00000000.00000000.00000000

Range - 224 to 239

class E



11110000.00000000.00000000.00000000

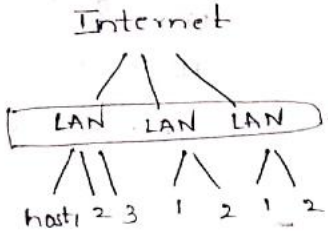
to

11111111.00000000.00000000.00000000

Range 240 to 255.

class	leading bits	Range of ip address
A	0	0 to 127 ⇒ 7 - N/w id, 24 - host id
B	10	128 to 191 ⇒ 14 - N/w id, 16 - host id
C	110	192 to 223 ⇒ 21 - N/w id, 8 - host id
D	1110	224 to 239 → Multicast
E	1111	240 to 255 → Research & Experiments.

Subnet Mask



for subnet making N/w bits- host bits-

class A - 8 bits - 24 bits — 11111111.00000000.00000000.00000000
N/w id host id
255.0.0.0

class B - 16 bits - N/w id — 11111111.11111111.00000000.00000000
16 bits - host id
255.255.0.0

class C - 24 bits - N/w id — 11111111.11111111.11111111.00000000
8 bits - host id
255.255.255.0

0 → represents N/w
255 → Broadcast purpose

in each & every class.

0 to 255 ⇒ 256

class A - 0 to 127

B - 128 to 191

C - 192 to 223

IP - 192.10.15.5

Subnet mask - 255.255.255.0

N/w id - ?

	128	64	32	16	8	4	2	1
192 -	1	1	0	0	0	0	0	0
10 -	0	0	0	0	1	0	1	0
15 -	0	0	0	0	1	1	1	1
5 -	0	0	0	0	0	1	0	1

IP address

11000000.00001010.

00001111.00000101

Subnet Mask-

11111111.11111111.

11111111.00000000

(AND) 11000000.00001010.

Operation 00001111.00000000

1 & 1 - 1

0 & 1 - 0

1 & 0 - 0

0 & 0 - 0

192.10.15.0 — N/w id

192.10.15.255 — Broadcast } 254 hosts.

In large N/w, some are divided into subnets.

Some hosts will acts as N/w id.

CIDR - Classless Inter Domain Routing

CIDR is a way to allocate & specify the internet addresses used in interdomain routing more flexible than with original system of i

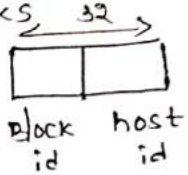
like assume use need ip for a min. no. of hosts of 300 under classful addressing you will be given class B which can accomodate more than 64K hosts. Hence CIDR can resolve this

class A → allows 128 N/w, 16 million hosts

B → 16,384 " , 64K "

C → " 2 million " , 256 K " .

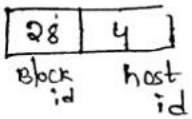
Here we divide in to blocks



Notation

x.y.z.w/n ← represent mask or no. of bits represent block

Ex: 200.10.20.40/28



host id = 4

No. of hosts = 2^4 = 16

Subnet mask will be represented by 28 with '1'.

11111111.11111111.11111111.11110000

255.255.255.240

Ex: 200.10.20.0010|1000

32 bit, 28, 0000

32 make to '0' to know n/w id.

200.10.20.32/28 → N/w id.

Protocols :-

Set of rules that are established to process data.

→ Different protocols are used for different processes.

ARP - Address Resolution Protocol.

IP address (N/w) ⟷ MAC address (System) (Mapping ip to MAC)

SLIP - Serial Line Internet Protocol (Packet formation)

Sequence of characters grouping → Packets

IPV4 - Internet Protocol. (Routing) → TCP/IP

IPV6 - updated of V4 (Works efficiently than IPV4)

(32 to 128 bits)

ICMP - Internet Control Message Protocol

(N/w devices will send error msgs in the n/w topology)

TCP - Transmission Control protocol

- Acknowledgement → Error checking is done
- Retransmission is done if pckt is lost
- Connection oriented.

UDP - User Datagram protocol, Connectionless

No Reliability, No error checking, No retransmission

RPC - (Remote Procedure Call)

- Connect with other device.

DHCP - Dynamic Host Configuration Protocol

- Distribute the ip address to new system in the n/w.

DNS - Domain Name System Protocol.

- Conversion of url to IP address.

FTP - File Transfer Protocol (file sharing b/w devices)

HTTP - Hyper Text Transfer Protocol.

(Communicate from client to server)

Browser ↓ ↑ Web server

Request - Response - Protocol.

HTTPS - Secure

https://www.gogle.com

IMAP - Internet Message Access Protocol

Email Protocol

Access/manipulate emails

Retrieve emails from mail servers

POP - Post Office Protocol

- Email Protocol
- Enables users to download emails from mail server to client

SMTP - Simple mail Transfer Protocol

Email Protocol. - Transfer Email from 1 user to another

Telnet - used to Connect remote devices
- enables admin to connect with any remote device with IP / host name

SNMP - Simple N/w Managing Protocol
- enables admin to monitor n/w Performance
- managing nodes.

UNIT - IV

Transport Layer

Congestion control:- congestion in the state in which n/w performance decreases. It happens when a n/w is holding too much packets which are more that the n/w capacity.

The n/w & transport layer share the responsibilities to handle congestion.

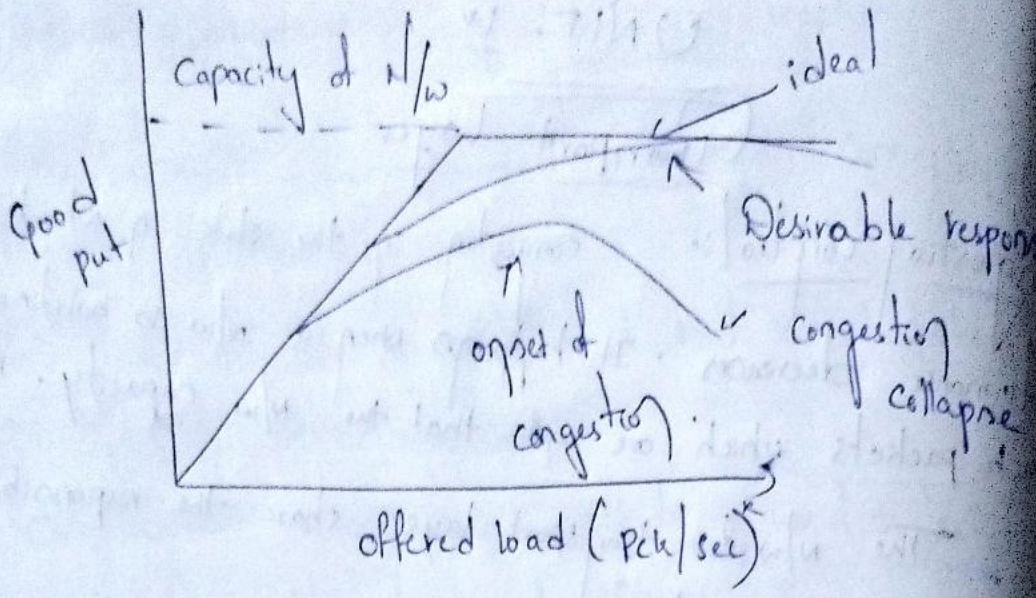
Congestion control → Routing was main functionality of n/w layer & second functionality is congestion control.

Approaches to congestion control:-

- → N/w provisioning
- → Traffic - Aware routing
- → Admission control
- → Traffic throttling
- → Load shedding

Congestion:- when two many packets are present in the Network - performances degrades.

* Both Network & Transport layer has the responsibility to control congestion.
* Network Services
  - connectionless — Congestion is not controlled. It is left to Transport layer
  - connection oriented — Congestion is controlled



→ when too much traffic is offered, congestion sets in & performance degrades sharply.

Congestion occured due to :-

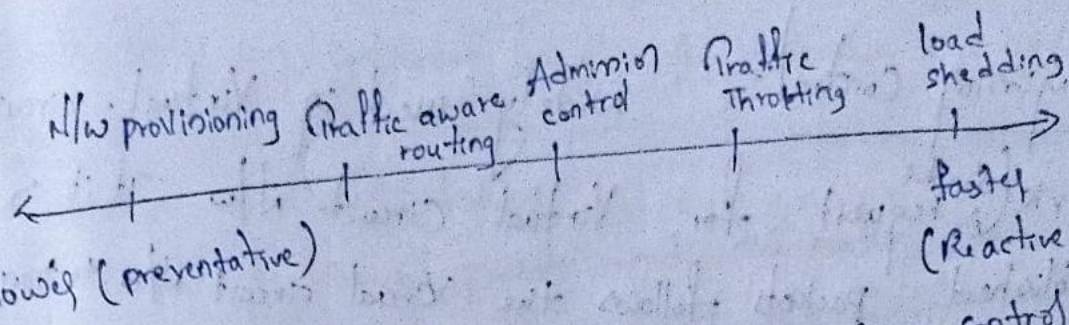
* Insufficient memory in routers Eg: queue is full.
* Slow processors Eg: CPU, routers are slow at processing
* Low Bandwidth lines.

Congestion control Vs Flow control

Congestion control :- Making sure that the entire subject is able to carry the offered traffic
↓
Routers & hosts will face the traffic

Flow control :- It is b/w Source & corresponded receivers that is the path b/w Source & Destination.

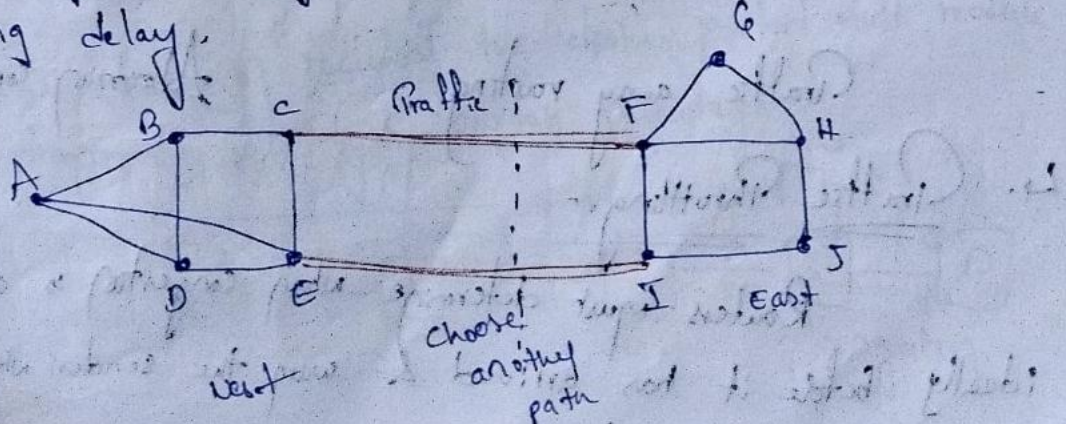
Approches to congestion control is either increases resources or decreases the load.



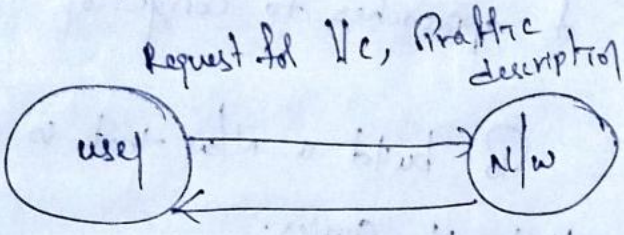
Time Scales of approaches to congestion control

1. N/w provisioning :- To build a N/w that is well matched to traffic that it carries.
   - Spare routers that are normally used only as backups.
   - Bandwidth on open market need to be purchased.
   - links and routers that are regularly heavily utilized are upgraded.

2. Traffic-aware Routing :- Routers can be tailored to traffic patterns - that change during the day as N/w users wake & sleep in d/f time zones.
   - Splitting traffic across multiple paths.
   - Set the link weight to be a function of link bandwidth & propagation delay plus the measured load or average queuing delay.



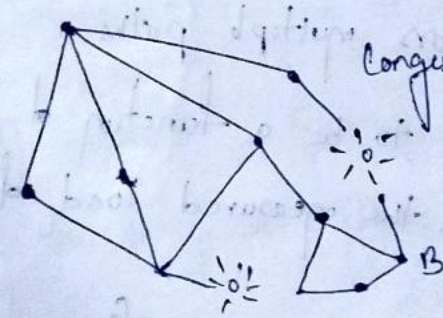
3. Admission control:- widely used in Virtual-circuit Network (Sender request for Virtual Circuit N/w. If it is established packets follows the Virtual circuit N/w to maintain)



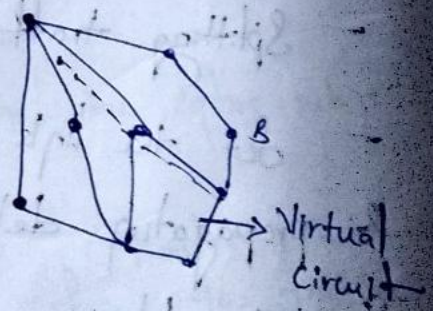
Request granted if the traffic can be accomodated without congestion, else denied

* It characterize traffic
* Revise enough capacity along the paths of each of its Virtual Circuits.

Combining admission control with Traffic aware routing:



Traffic away routing



Admission control

4. Traffic Throttling:-

Routers must determine when congestion is approaching ideally before it has arrived & warn the sender who is making this congestion.

* Each router can continously monitor the resources it is using.

Resources:

1. utilization of output links
2. Buffering of queued packets inside the router
3. No. of packets that are lost due to insufficient Buffering.

Estimation of the queuing delay (EWMA (exponentially weighted moving Avg)

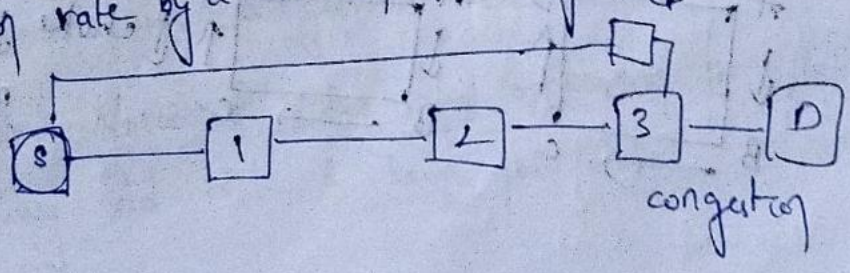
$d_{new} = \alpha d_{old} + (1-\alpha) s$

nearest feature. previous history

d = estimate of the queuing delay.
s = A sample of instantaneous queue length.
α = Determines how fast the router forgets recent history.
α value should be b/w 0 & 1

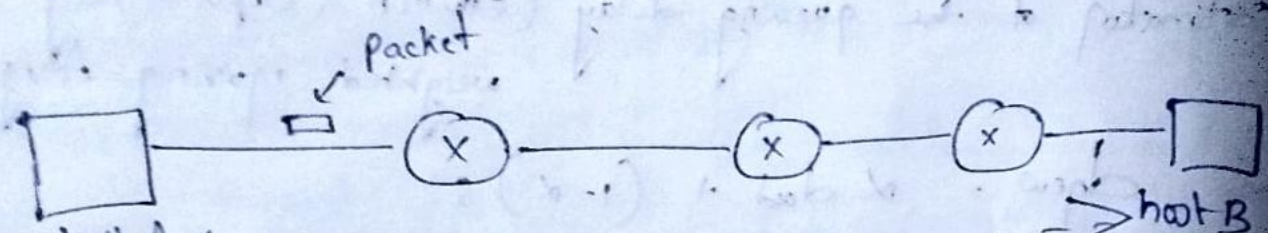
a) Choke packet:- A more direct way of telling the source to slow down.

- A choke packet is a control packet generated at a congested node & transmitted to restrict traffic flow.
- The source on receiving the choke packet must reduce its transmission rate by a certain percentage.



b) Explict congestion Notification:-

- using warning bits in packet header.
- Destination notes that there is congestion & inform the sender in its reply.



Congestion signal

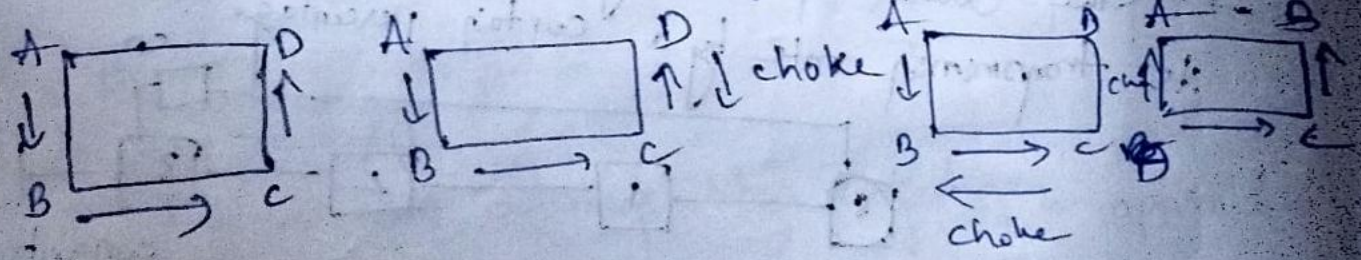
c) Hop by Hop back pressure

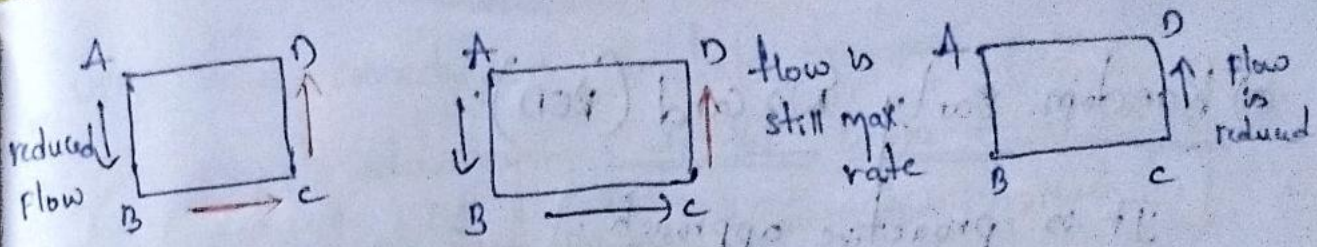
At high speeds & over long distances, sending a choke packet to source hosts doesn't work well because reaction is slow.

Alternative approach:-

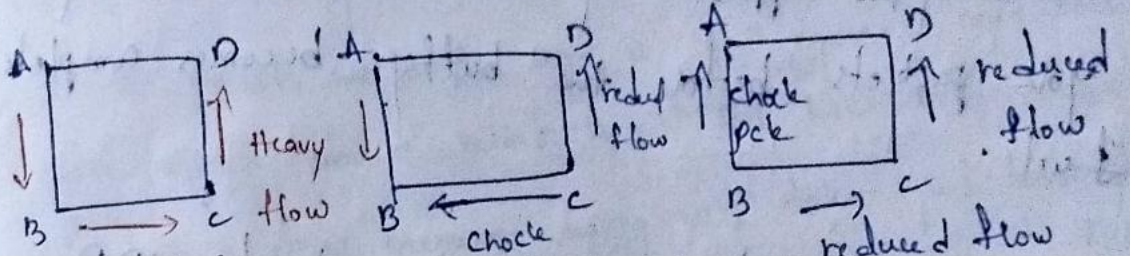
Do have choke packet take effected at every hop it passes through.

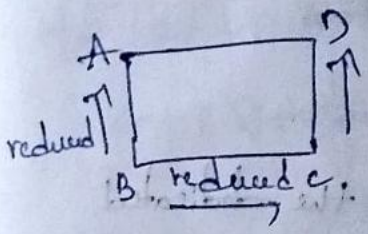
Direct choke packets: Hop by Hop Back pressure





(a) choke packet that affects only source





(b) A choke packet that affects each hop it passes through.

5) Load Shedding — It is one of the most useful one in congestion occurance.

- when buffer becomes full routers simply discard pcks.
- which packets is choosen to be the victim depends on the application & on the error strategy used in data-link layer.

Eg: 1. For a file transfer, we cannot discard older pck since that will cause a gap in received data.

2. For real time Voice & Video chat application it is better to through away old data & keep new data.

a) Random Early Discard (RED):

It is proactive approach.

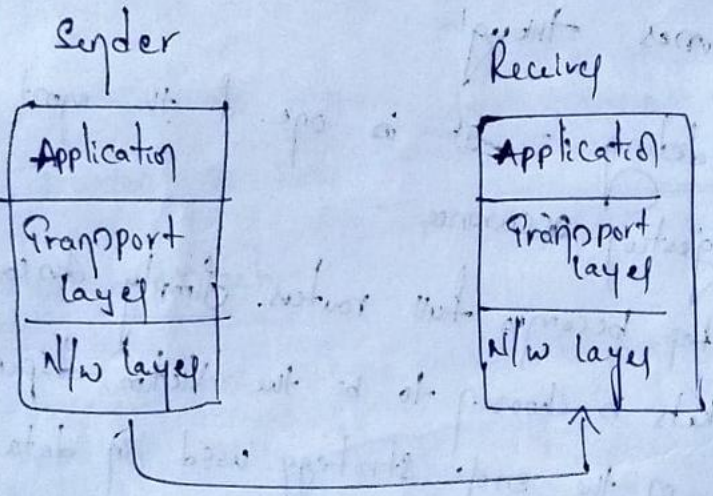
→ congestion prevention

→ Here in this approath in which router discards one & more packets before the buffer becomes completely full.

UDP: Connectionless transport protocol :- UDP

↳ User Datagram protocol

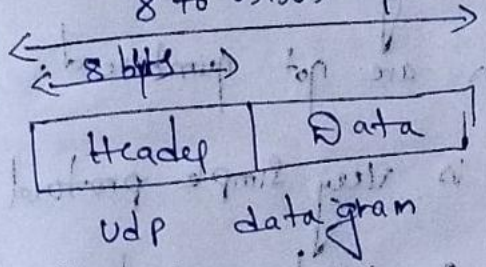
Unlike TCP, UDP doesn't establish the dedicated connection b/w sender & receiver



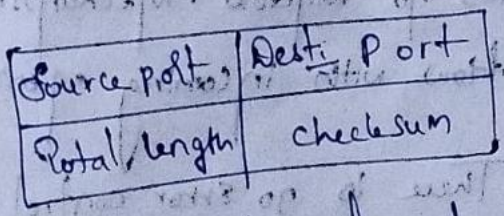
using UDP as Transport layer protocol in Transport layer receives message from application layer & send to network layer from their it send to N/w layer at receiver side

* UDP is a connectionless, unreliable transport protocol.
* It doesn't add anything to the service of IP except for providing process-to-process communication instead of host-to-host communication.
* UDP is a very simple protocol using minimum overhead.
* If a process wants to send a small message and doesn't care much about reliability, it can use UDP.
* Sending a small message using UDP takes much less interaction b/w sender & receiver than using TCP.

UDP Datagram:



UDP datagram



Header format

* UDP packet called user datagrams, have a fixed size header of 8 bytes made of four fields, each of 2 bytes (16 bits).
* The first two fields define the source & destination port no's.
* The third field defines the total length of user datagram header plus data. The 16 bits can define a total length 0 to 65,535 bytes.
* The last field can carry the optional check sum.

UDP Services:

Process to process communication: UDP provides P-to-P communication using socket address, a combination of IP & port no's.

Connectionless Services: It provides connectionless services. This means that each datagram sent by UDP is a independent datagram.

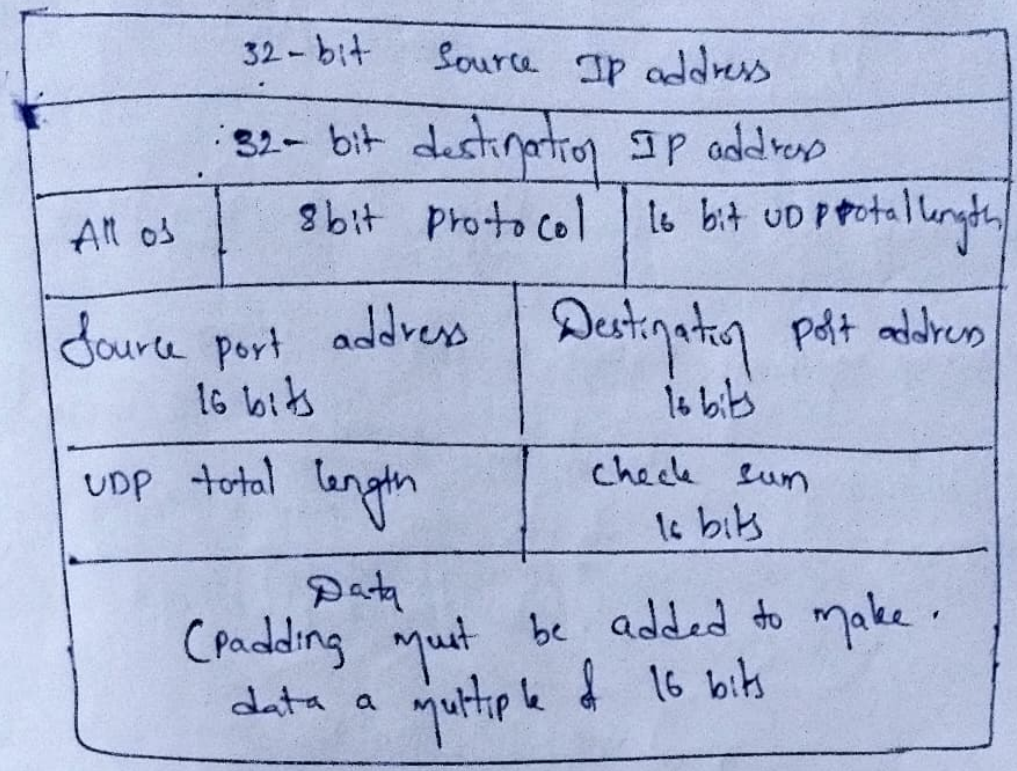
* There is no relationship b/w different user datagram even if they are coming from the same source process & going to same destination program.
* The user datagram are not numbered.

Flow control: UDP is very simple protocol. There is no flow control & hence no window mechanism. The receiver may overflow with incoming messages.

Error control: There is no error control mechanism in UDP except for checksum. This means that the sender doesn't know if a message has been lost or duplicated.

→ When the receiver detects an error through the checksum, the user datagram is silently discarded.

Check Sum:-


32-bit Source IP address
32-bit destination IP address
All 0s
8 bit protocol
16 bit UDP total length
Source port address 16 bits
Destination port address 16 bits
UDP total length
Check sum 16 bits
Data
(padding must be added to make data a multiple of 16 bits

COMPUTER NETWORKS

UNIT 5

SYLLABUS: The Transport Layer: Transport layer protocols: Introduction-services- port number-User data gram protocol-User datagram-UDP services-UDP applications-Transmission control protocol: TCP services- TCP features- Segment- A TCP connection- windows in TCP- flow control-Error control, Congestion control in TCP.

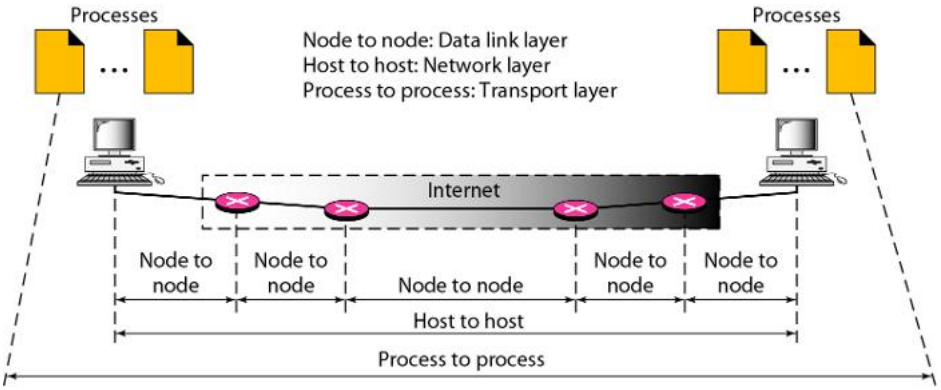
Application Layer - World Wide Web: HTTP, Electronic mail-Architecture- web based mail- email security- TELENET-local versus remote Logging-Domain Name System: Name Space, DNS in Internet - Resolution-Caching- Resource Records- DNS messages- Registrars-security of DNS Name Servers, SNMP.

5.1 Transport layer services:

5.1.1 PROCESS-TO-PROCESS DELIVERY:

The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship.

The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called *node-to-node delivery.* The network layer is responsible for delivery of datagrams between two hosts. This is called *host-to-host delivery.* Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts. Real communication takes place between two processes (application programs). We need process-to-process delivery. However, at any moment, several processes may be running on the source host and several on the destination host. To complete the delivery, we need a mechanism to deliver data from one of these processes running on the source host to the corresponding process running on the destination host.

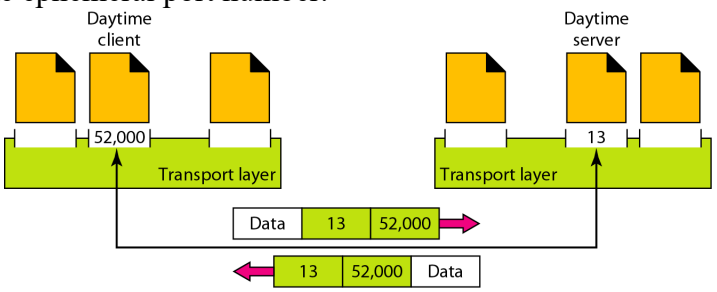


5.1.2 PORT NUMBERS:

At the network layer, we need an IP address to choose one host among millions. A datagram in the network layer needs a destination IP address for delivery and a source IP address for the destination's reply.

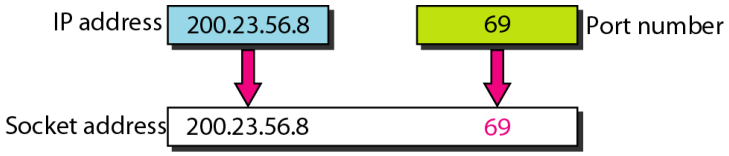
At the transport layer, we need a transport layer address, called a port number, to choose among multiple processes running on the destination host. The destination port number is needed for delivery; the source port number is needed for the reply.

In the Internet model, the port numbers are 16-bit integers between 0 and 65,535. The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the ephemeral port number.



Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.

A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address. These four pieces of information are part of the IP header and the transport layer protocol header. The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.



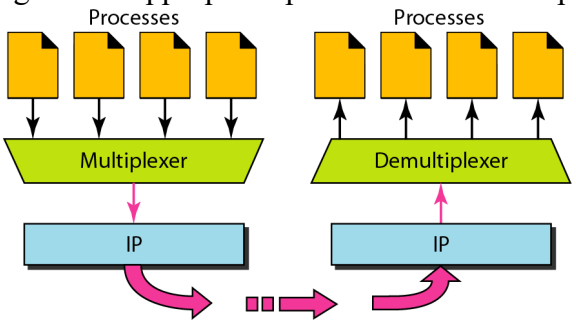
5.1.3 MULTIPLE ING AND DEMULTIPLE ING

Multiplexing

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing. The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing

At the receiver site, the relationship is one-to-many and requires demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.



5.1.4 CONNECTIONLESS VERSUS CONNECTION-ORIENTED SERVICE

A transport layer protocol can either be connectionless or connection-oriented.

Connectionless Service

In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. We will see shortly that one of the transport layer protocols in the Internet model, UDP, is connectionless.

Connection riented Service

In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released. We will see shortly that TCP and SCTP are connection-oriented protocols.

5.1.5 RELIABLE VERSUS UNRELIABLE

The transport layer service can be reliable or unreliable. If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service. On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.

In the Internet, there are three common different transport layer protocols, as we have already mentioned. UDP is connectionless and unreliable; TCP and SCTP are connection oriented and reliable. These three can respond to the demands of the application layer programs.

5.2 USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs very limited error checking.

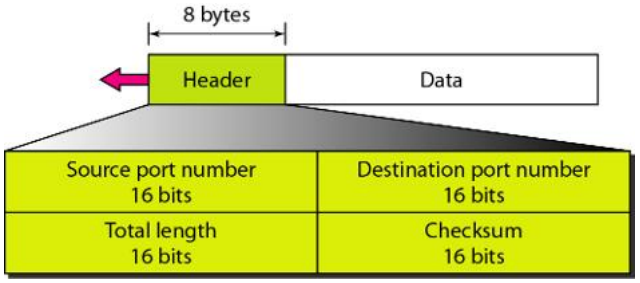
5.2.1 Well-Known Ports for UDP

Below table shows some well-known port numbers used by UDP. Some port numbers can be used by both UDP and TCP.

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

5.2.2 User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. Below Figure shows the format of a user datagram.



The fields are as follows:

- **Source port number.** This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server Sending a response), the port number, in most cases, is a well-known port number.
- **Destination port number.** This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.
- **Length.** This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.
- **Checksum**. This field is used to detect errors over the entire user datagram (header plus data). The checksum is discussed next.

5.2.3 UDP Operation (Services)

UDP uses concepts common to the transport layer. These concepts will be discussed here briefly, and then expanded in the next section on the TCP protocol.

. *Connectionless Services*

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

2. *Flow and Error Control*

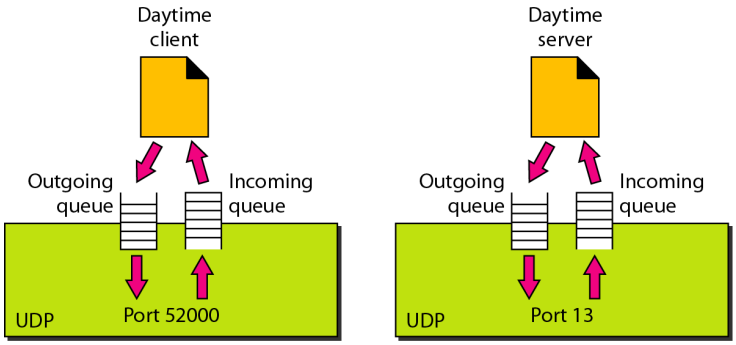
UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages. There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.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

. *Encapsulation and Decapsulation*

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

. *Queuing*

We have talked about ports without discussing the actual implementation of them. InUDP, queues are associated with ports.



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

Note that even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue. The queues opened by the client are, in most cases, identified by ephemeral port numbers. The queues function as long as the process is running. When the process terminates, the queues are destroyed.

The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow. If this happens, the operating system can ask the client process to wait before sending any more messages.

5.2.4 UDP applications

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. It is not usually used for a process such as FTP that needs to send bulk data.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP).

5.3 Transmission Control Protocol (TCP)

TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.

5.3.1 TCP Services

Before we discuss TCP in detail, let us explain the services offered by TCP to the processes at the application layer.

. *Process-to-Process Communication*

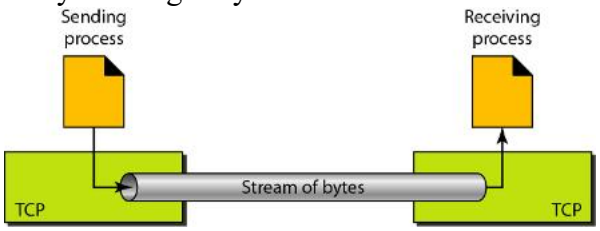
Like UDP, TCP provides process-to-process communication using port numbers. Below Table lists some well-known port numbers used by TCP.

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

2. *Stream Delivery Service*

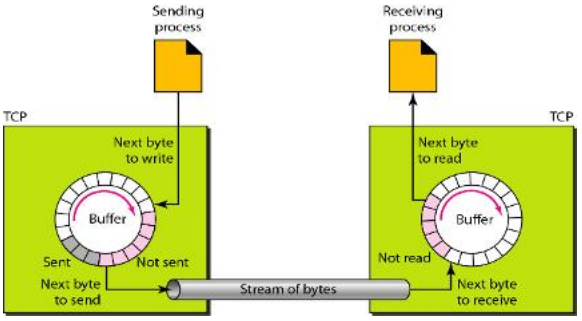
TCP, unlike UDP, is a stream-oriented protocol. In UDP, a process (an application program) sends messages, with predefined boundaries, to UDP for delivery. UDP adds its own header to each of these messages and delivers them to IP for transmission. Each message from the process is called a user datagram and becomes, eventually, one IP datagram. Neither IP nor UDP recognizes any relationship between the datagrams.

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



. *Sending and Receiving Buffers*

The sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. (We will see later that these buffers are also necessary for flow and error control mechanisms used by TCP.) One way to implement a buffer is to use a circular array of I-byte locations as shown in below Figure. For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.



Above Figure shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP. However, as we will see later in this chapter, TCP may be able to send only part of this colored section. This could be due to the slowness of the receiving process or perhaps to congestion in the network. Also note that after the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process. This is why we show a circular buffer.

. *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.

5. *Connection- riented Service*

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

- The two TCPs establish a connection between them.
- Data are exchanged in both directions.
- The connection is terminated.

. *Reliable Service*

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

5.3.2 TCP Features

. *Numbering System*

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

Byte Number TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. The numbering does not necessarily start from O. Instead, TCP generates a random number between 0 and 232 - 1 for the number of the first byte.

For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

2. *Flow Control*

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

. *Error Control*

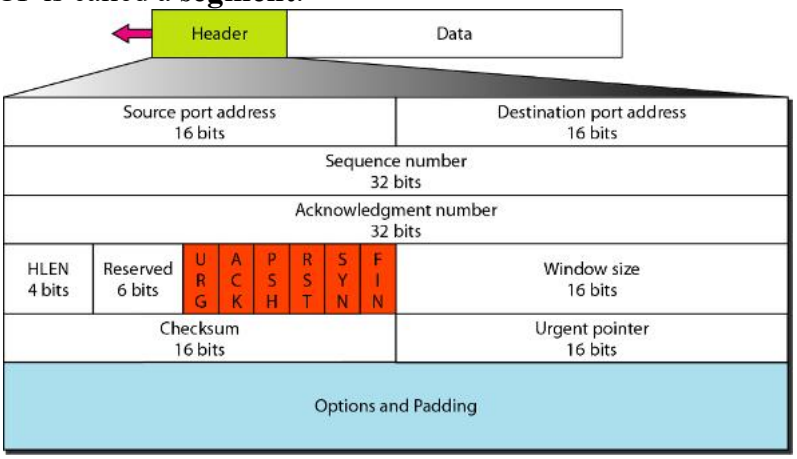
To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented, as we will see later.

. *Congestion Control*

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

5.3.3 Segment

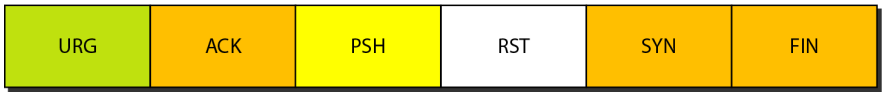
A packet in TCP is called a **segment**.



The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. We will discuss some of the header fields in this section.

- **Source port address**. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.
- **Destination port address.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.
- **Sequence number**. This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.
- **Acknowledgment number.** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x* + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- **Header length**. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- **Reserved**. This is a 6-bit field reserved for future use.
- **Control**. This field defines 6 different control bits or flags as shown in below figure. One or more of these bits can be set at a time.

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



These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in below table.

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

- **Window si e.** This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- **Checksum**. This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.
- **Urgent pointer**. This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- **Options**. There can be up to 40 bytes of optional information in the TCP header.

5.3.4 A TCP Connection

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

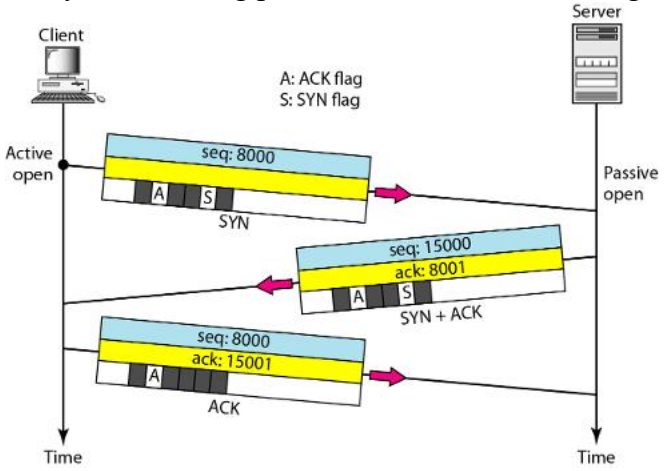
In TCP, connection-oriented transmission requires three phases: **connection establishment, data transfer, and connection termination.**

. *Connection Establishment*

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

Three-Way Handshaking The connection establishment in TCP is called three-way handshaking.

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



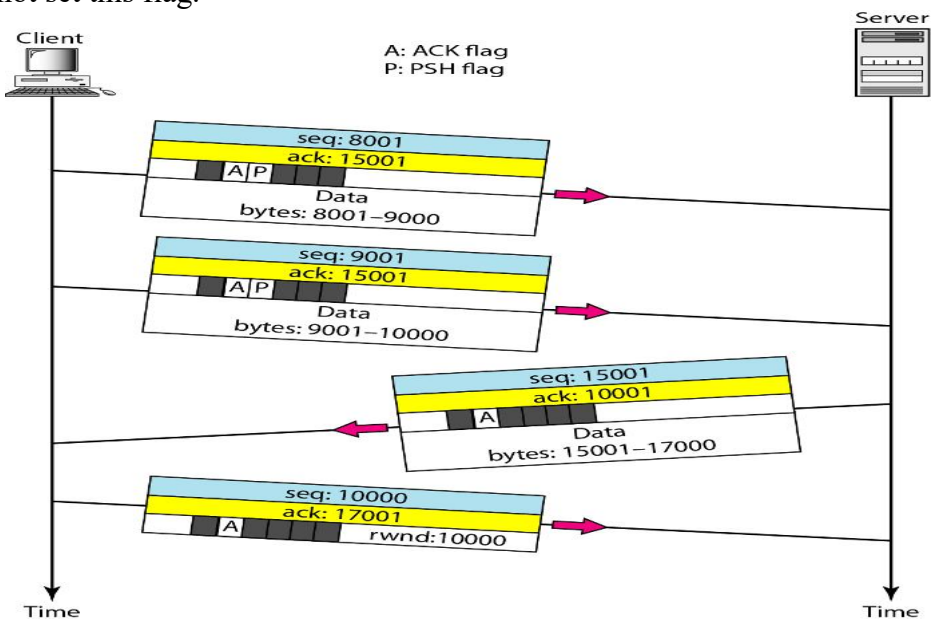
1. The client sends the first segment, a S N segment, in which only the S N flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the S N segment carries no real data,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

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

2. *Data Transfer*

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. for the moment, it is enough to know that data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data.

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

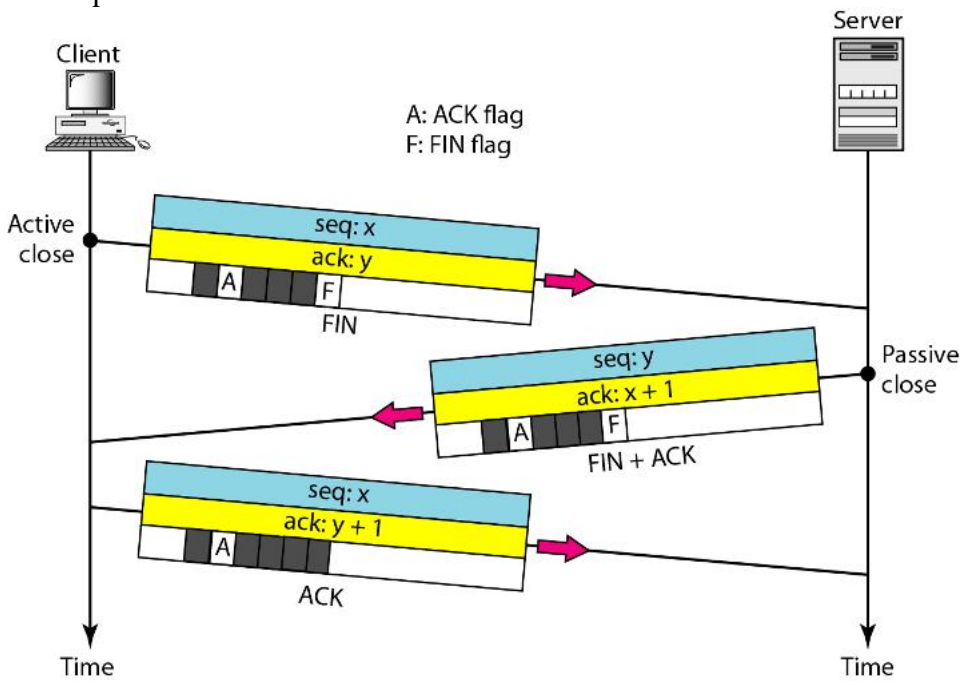


. *Connection Termination*

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

Three-Way Handshaking for connection termination as shown in below figure.

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in below Figure. If it is only a control segment, it consumes only one sequence number.
2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk f data from the server. If it does not carry data, it consumes only one sequence number.
3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.



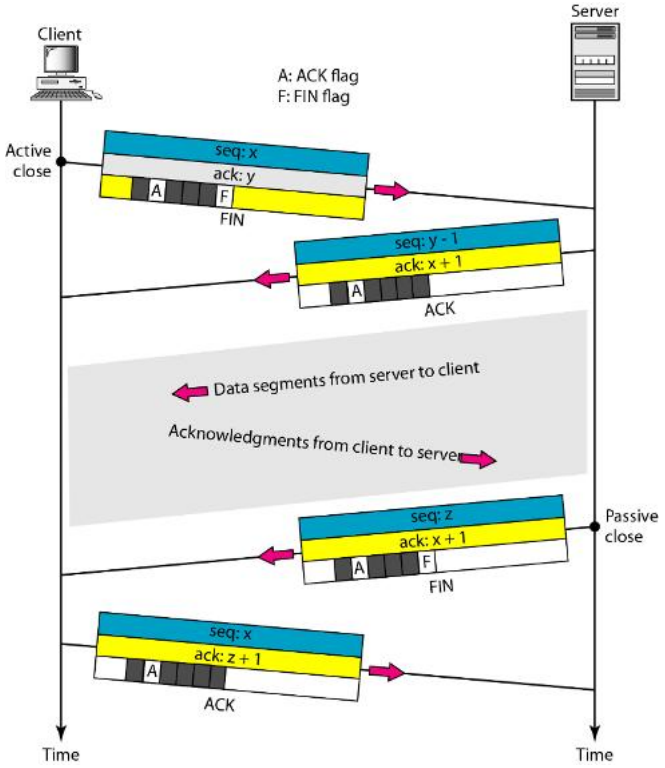
Half-Close in TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin. Below figure shows an example of a half-close.

The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops.

The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server.

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



5.3.5 Flow Control

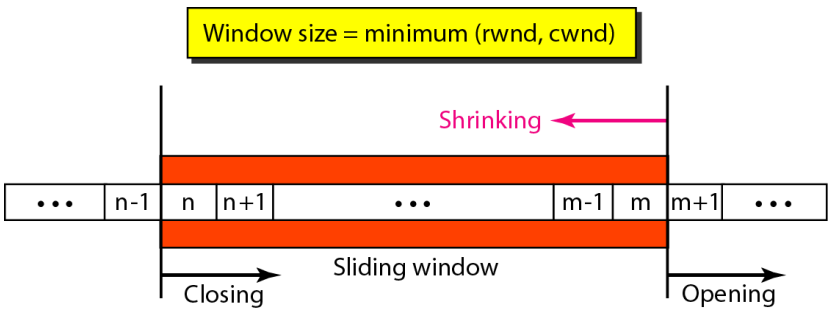
TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go-Back-N protocol because t does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

There are two big differences between this sliding window and the one we used at the data link layer. First, the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented. Second, the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size.

Below figure shows the sliding window in TCP. The window spans a portion of the buffer containing bytes received from the process. The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment. The imaginary window has two walls: one left and one right.

The window is *opened, closed,* or *shrunk.* These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore. Shrinking the window means moving the right wall to the left. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending.



Some points about TCP sliding windows:

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

5.3. Error Control

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

. *Checksum*

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

2. *Acknowledgment*

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

. *Retransmission*

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

5.3. Congestion Control in TCP

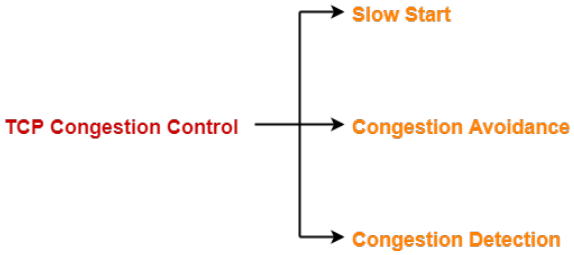
The size of the sender window is determined by the following two factors-

1. **Receiver window si e**
2. **Congestion window si e**

1. **Receiver Window Si e:**
   - Sender should not send data greater than receiver window size.
   - Otherwise, it leads to dropping the TCP segments which causes **TCP Retransmission**.
   - So, sender should always send data less than or equal to receiver window size.
   - Receiver dictates its window size to the sender through **TCP Header**.

- Sender should not send data greater than congestion window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to congestion window size.
- Different variants of TCP use different approaches to calculate the size of congestion window.
- Congestion window is known only to the sender and is not sent over the links.

TCP's general policy for handling congestion consists of following three phases-



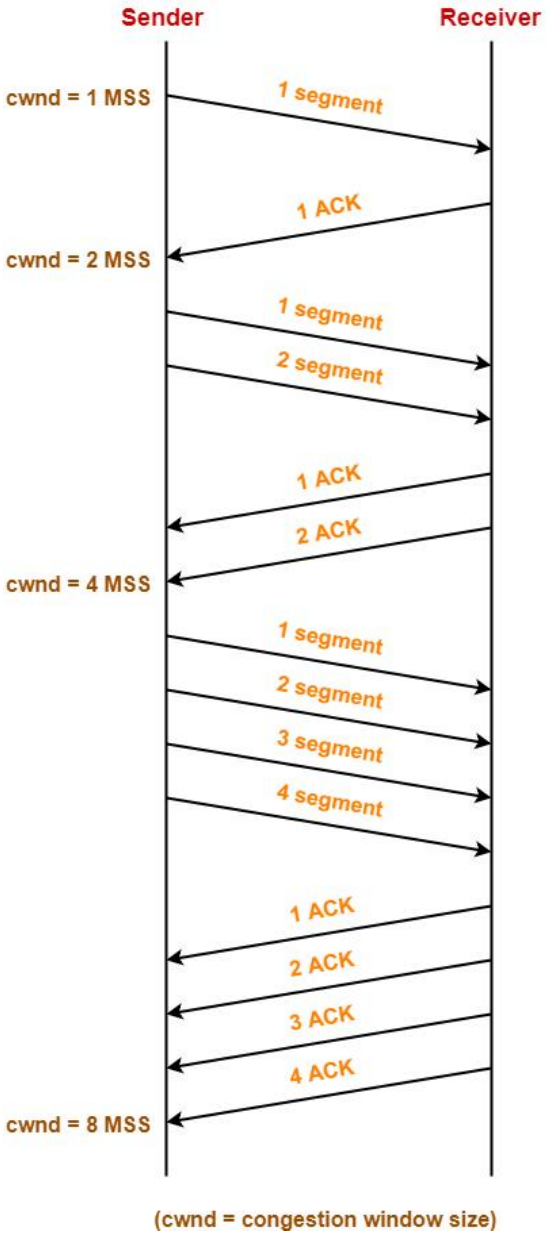
1. Slow Start Phase-

- Initially, sender sets congestion window size = Maximum Segment Size (1 MSS).
- After receiving each acknowledgment, sender increases the congestion window size by 1 MSS.
- In this phase, the size of congestion window increases exponentially.

The followed formula is-

Congestion window size = Congestion window size + Maximum segment size

This is shown below-



- After 1 round trip time, congestion window size = $(2)^1 = 2$ MSS
- After 2 round trip time, congestion window size = $(2)^2 = 4$ MSS
- After 3 round trip time, congestion window size = $(2)^3 = 8$ MSS and so on.

This phase continues until the congestion window size reaches the slow start threshold.

Threshold

= Maximum number of TCP segments that receiver window can accommodate / 2

= (Receiver window size / Maximum Segment Size) / 2

2. Congestion Avoidance Phase-

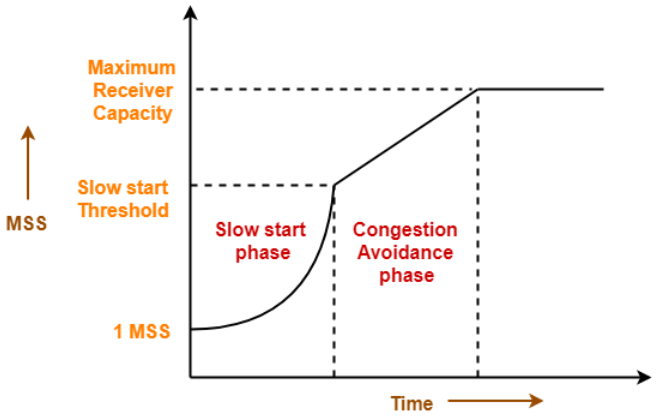
After reaching the threshold,

- Sender increases the congestion window size linearly to avoid the congestion.
- On receiving each acknowledgement, sender increments the congestion window size by 1.

The followed formula is-

Congestion window size = Congestion window size + 1

This phase continues until the congestion window size becomes equal to the receiver window size.



When sender detects the loss of segments, it reacts in different ways depending on how the loss is detected-

- Time Out Timer expires before receiving the acknowledgement for a segment.
- This case suggests the stronger possibility of congestion in the network.
- There are chances that a segment has been dropped in the network.

In this case, sender reacts by-

- Setting the slow start threshold to half of the current congestion window size.
- Decreasing the congestion window size to 1 MSS.
- Resuming the slow start phase.

- Sender receives 3 duplicate acknowledgements for a segment.
- This case suggests the weaker possibility of congestion in the network.
- There are chances that a segment has been dropped but few segments sent later may have reached.

In this case, sender reacts by-

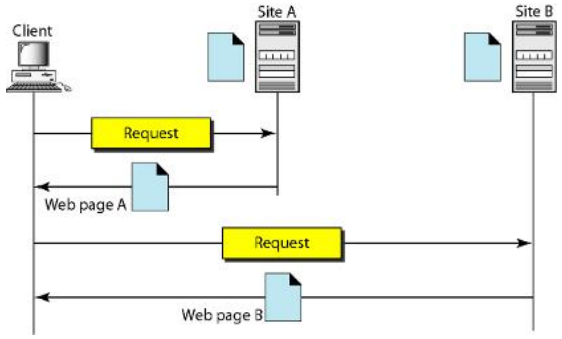
- Setting the slow start threshold to half of the current congestion window size.
- Decreasing the congestion window size to slow start threshold.
- Resuming the congestion avoidance phase.

5.4 World Wide Web (WWW)

The **World Wide Web** (WWW) is a repository of information linked together from points all over the world. TheWWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research.

5.4.1 ARCHITECTURE

The WWW today is a distributed clientJserver service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites,* as shown in Figure.

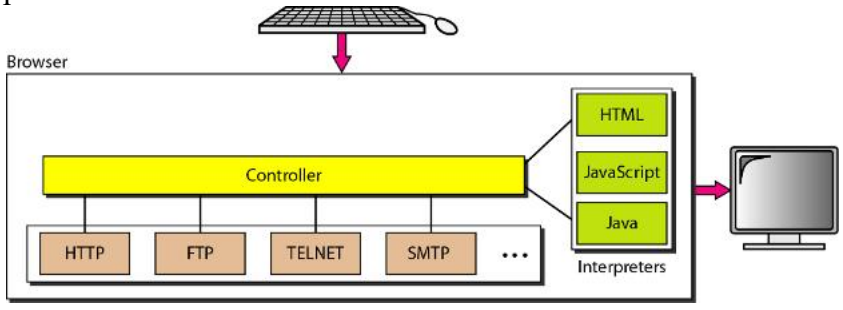


Each site holds one or more documents, referred to as *Web pages.* Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers.

The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL. The server at site A finds the document and sends it to the client. When the user views the document, she finds some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

5.4.2 Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as or HTIP. The interpreter can be HTML, ava, or JavaScript, depending on the type of document.

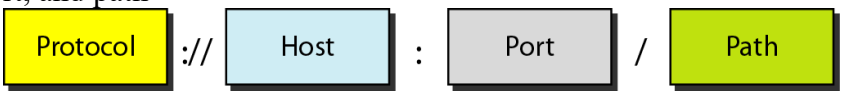


5.4.3 Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk.

5.4.4 Uniform Resource Locator

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



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

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.

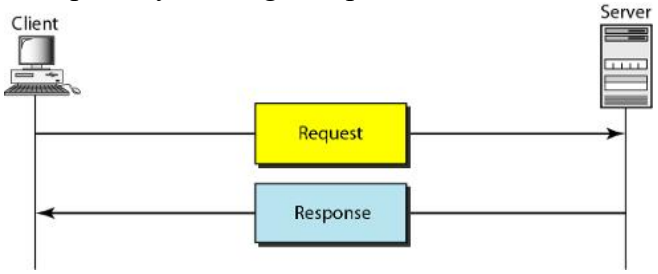
Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

5.5 HYPERTE T TRANSFER PROTOCOL (HTTP)

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. HTTP uses the services of TCP on well-known port 80.

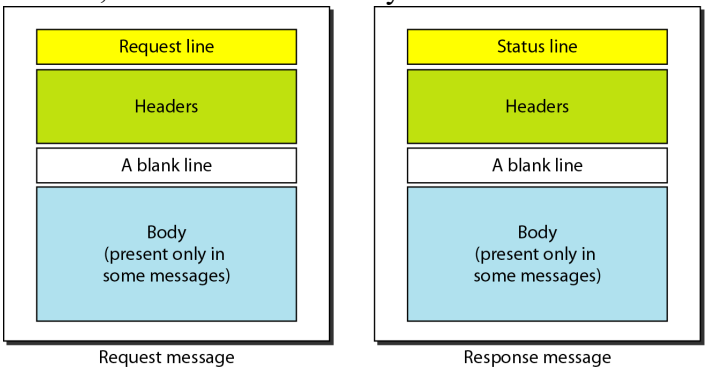
5.5.1 HTTP Transaction

Below figure illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.

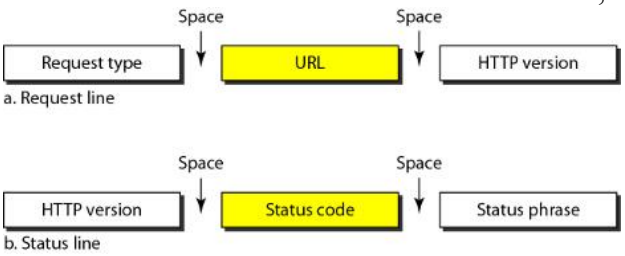


5.5.2 HTTP Messages

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



Request and Status Lines. The first line in a request message is called a request line; the first line in the response message is called the status line. There is one common field, as shown in below figure.



Request type. This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The request type is categorized into *methods* as defined in below table.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

URL. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet..

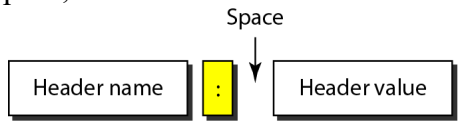
Version. The most current version of HTTP is 1.1.

Status code. This field is used in the response message. The status code field is similar to those in the FTP and the SMTP protocols. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request.

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

Code	*Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Header The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value.



A header line belongs to one of four categories: **general header, request header, response header, and entity header.**

General header The general header gives general information about the message and can be present in both a request and a response. Below table lists some general headers with their descriptions.

Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Request header The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format. See below Table for a list of some request headers and their descriptions.

Header	*Description*
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

Response header The response header can be present only in a response message. It specifies the server's configuration and special information about the request. See below Table for a list of some response headers with their descriptions.

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

Entity header The entity header gives information about the body of the document. Although it is mostly present in response messages, some request messages, such as POST or PUT methods, that contain a body also use this type of header. See below Table for a list of some entity headers and their descriptions.

Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

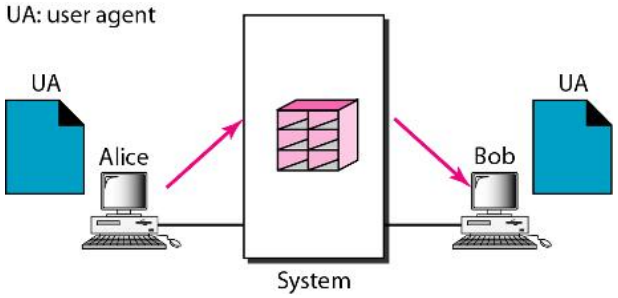
5. ELECTRONIC MAIL

One of the most popular Internet services is electronic mail (e-mail). At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

5. .1 Architecture

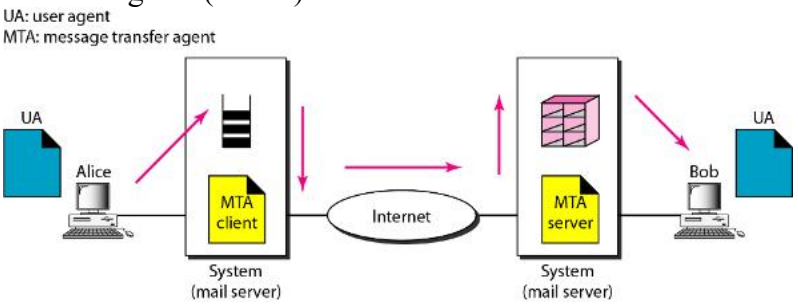
. *First Scenario*

In the first scenario, the sender and the receiver of the e-mail are users (or 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, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, 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 of files). Bob can retrieve and read the contents of his mailbox at his convenience, using a user agent.



2. *Second Scenario*

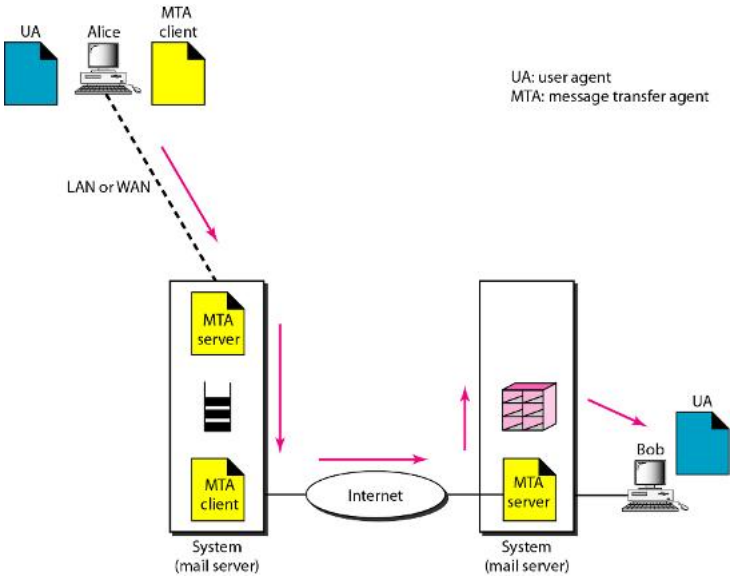
In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systems. The message needs to be sent over the Internet. Here we need user agents (UAs) and message transfer agents (MTAs).



Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her 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, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one 'client and one server. Like most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection. The client, on the other hand, can be alerted by the system when there is a message in the queue to be sent.

. *Third Scenario*

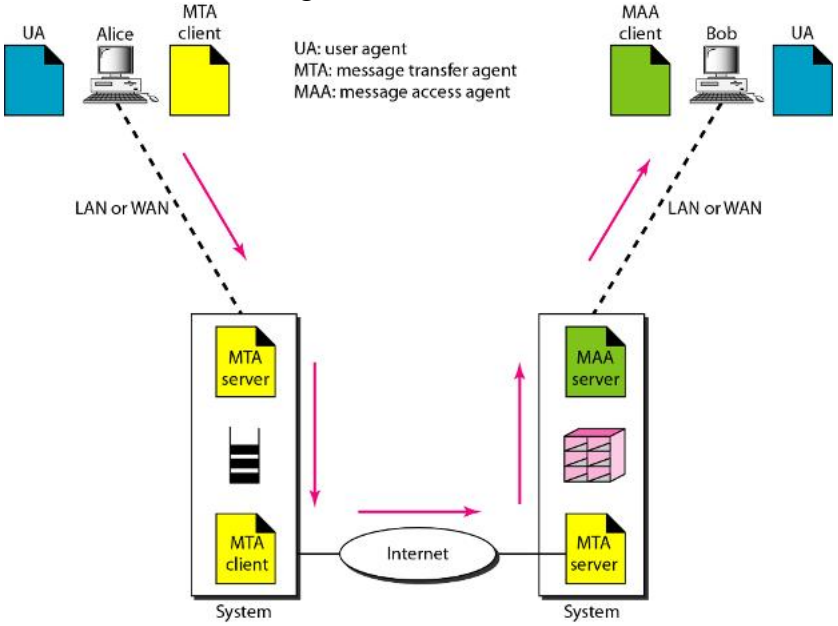
In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization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



Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in tum, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox. At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client/server programs.

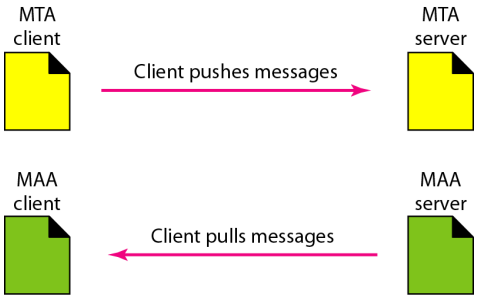
. *Fourth Scenario*

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it. Here, we need another set of client/server agents, which we call message access agents (MAAs). Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages.



There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a-WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

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

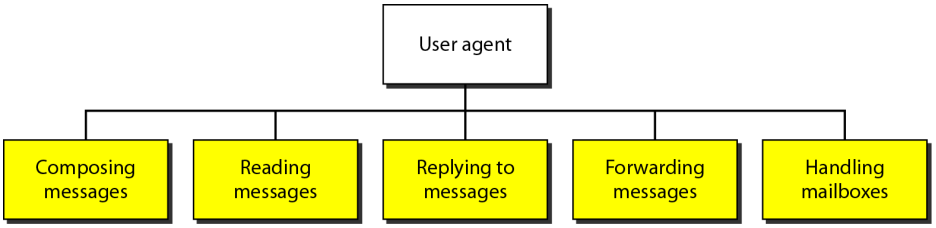


5. .2 User Agent

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

. *Services Provided by a User Agent*

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



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

Reading Messages The second duty of the user agent is to read the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Most user agents show a one-line summary of each received mail. Each e-mail contains the following fields.

1. A number field.
2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to.
3. The size of the message.
4. The sender.
5. The optional subject field.

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

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

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

2. *User Agent Types*

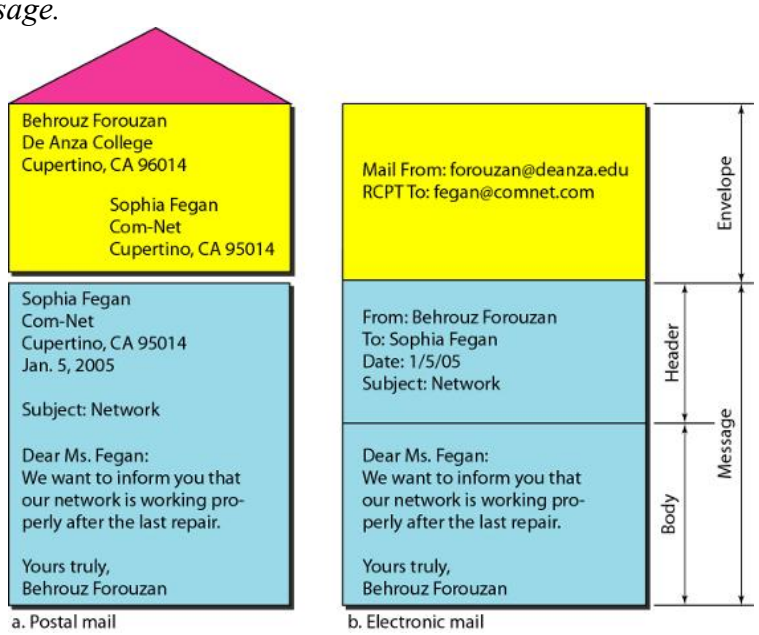
There are two types of user agents: command-driven and GUI-based.

Command-Driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients.

GUI-Based Modern user agents are GUI-based. They contain graphical-user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu bars, and windows that make the services easy to access. Some examples of GUI-based user agents are Eudora, Microsoft's Outlook, and Netscape.

. *Sending Mail*

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



Envelope

The envelope usually contains the sender and the receiver addresses. Message The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information (such as encoding type, as we see shortly). The body of the message contains the actual information to be read by the recipient.

Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the *UA* informs the user with a notice. If the user is ready to read the mail. A list is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

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

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

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

. *Mailing List*

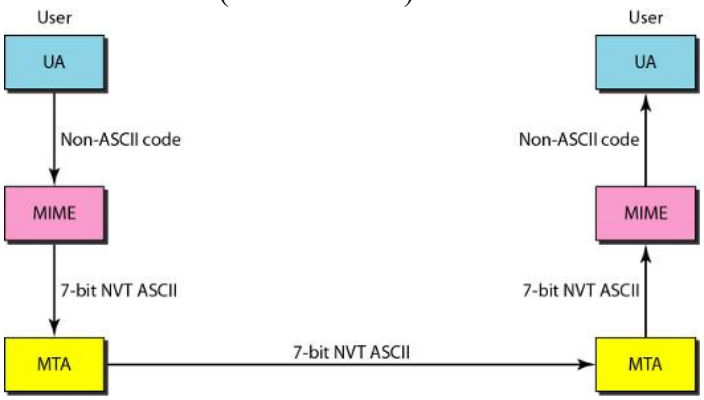
Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry

in the list, must be prepared and handed to the MTA. If there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity.

5. .3 MIME

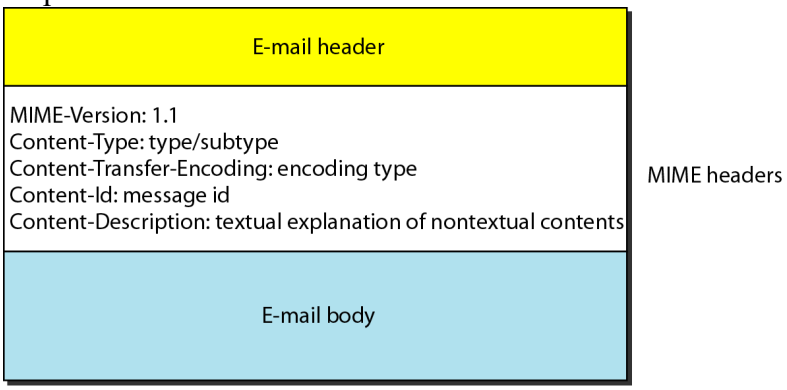
Electronic mail has a simple structure. Its simplicity, however, comes at a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data. We can think of MIME as a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa.



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

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



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

MIME-Version: 1.1

Content-Type This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters.

Content-Type: type subtype, parameter

MIME allows seven different types of data.

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

Content-Transfer-Encoding This header defines the method used to encode the messages into 0s and 1s for transport:

Content-Transfer-Encoding: type

Type	Description
7-bit	NVT ASCII characters and short lines
8-bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base-64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equals sign followed by an ASCII code

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

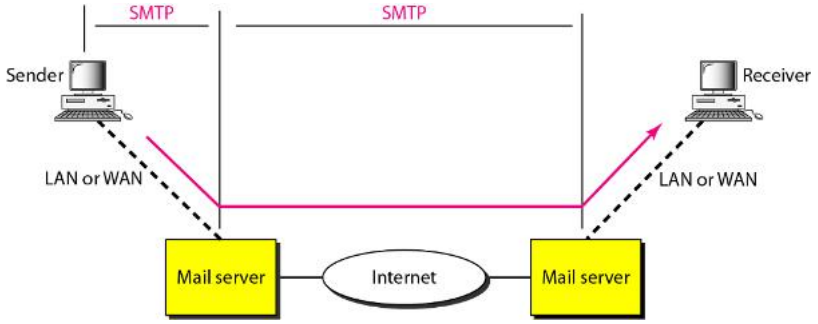
Content-Id: id content-id

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

Content-Description: description

5. .4 Message Transfer Agent: SMTP

The actual mail transfer is done through message transfer agents. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP). As we said before, two pairs of MTA client/server programs are used in the most common situation (fourth scenario).

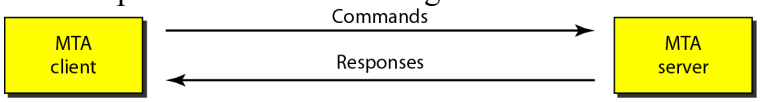


SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

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



Commands are sent from the client to the server. It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands. The first five are mandatory; every implementation must support these five commands. The next three are often used and highly recommended.

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

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

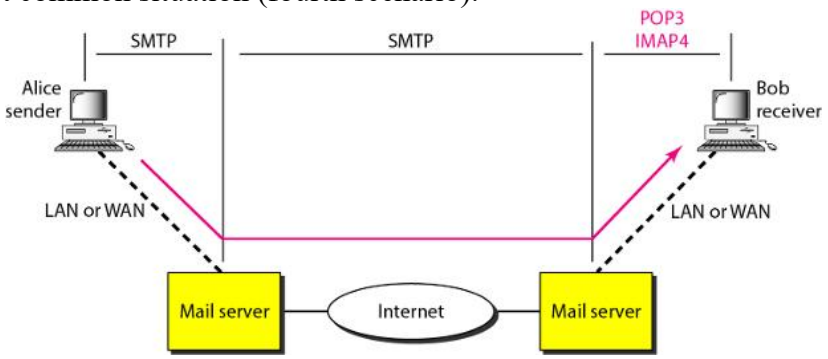
Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted: insufficient storage

5. .5 Message Access Agent: POP and IMAP

The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from the client to the server. In

other words, the direction of the bulk: data (messages) is from the client to the server. On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. The third stage uses a message access agent.

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



P P

Post Office Protocol, version 3 (POP3) is simple and limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

Mail access starts with the client when the user needs to download e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110.

It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one.

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

IMAP

Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4)**. IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

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

5. . Web-Based Mail

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Two common sites are Hotmail and ahoo. The idea is very simple. Mail transfer from Alice's browser to her mail server is done through HTTP The transfer of the message from the sending mail server to the receiving mail server is still through SMTP. Finally, the message from the receiving server (the Web server) to Bob's browser is done through HTIP.

The last phase is very interesting. Instead of POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in

name and password match, the e-mail is transferred from the Web server to Bob's browser in HTML format.

5. TELNET

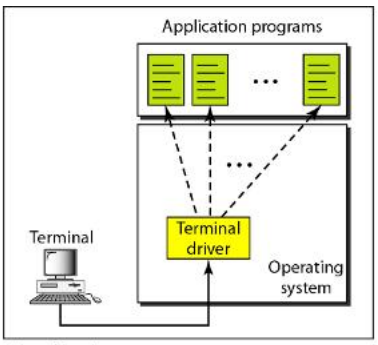
TELNET is an abbreviation for *TErminaL NETwork.* It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO). TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be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

Timesharing Environment

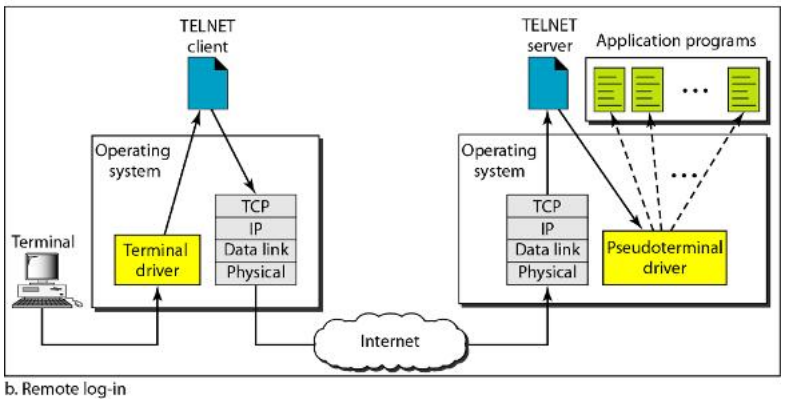
TELNET was designed at a time when most operating systems, such as UNIX, were operating in a timesharing environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard,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

Logging

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



a. Local log-in



b. Remote log-in

When a user logs into a local timesharing system, it is called local log-in. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

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

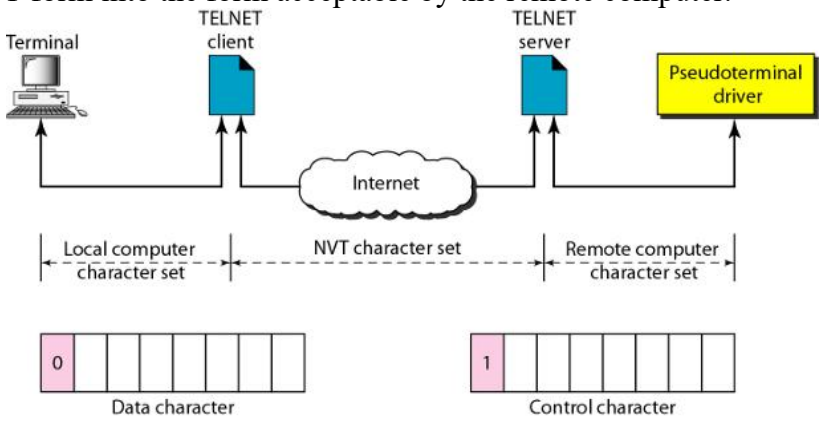
The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the

remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver.

Network Virtual Terminal

The mechanism to access a remote computer is complex. This is so because every computer and its operating system accept a special combination of characters as tokens. For example, the end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d.

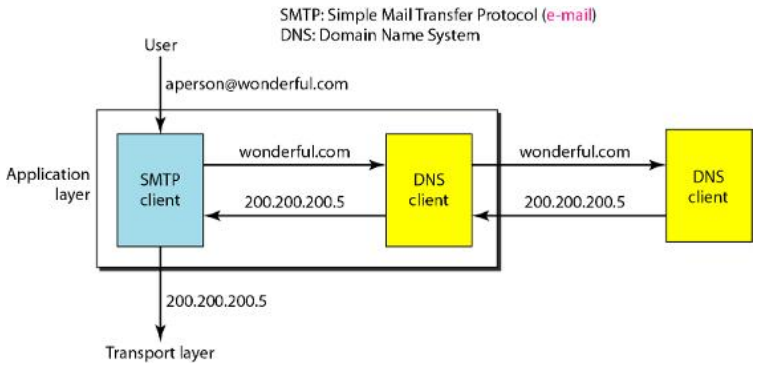
We are dealing with heterogeneous systems. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the network virtual terminal (NVT) character set. Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer.



Domain Name System

DNS stands for Domain Name System. DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.

Below figure shows an example of how 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 may know the e-mail address of the recipient; however, 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.



5. .1 NAME SPACE

A name space that maps each address to a unique name can be organized in two ways: fiat or hierarchical.

1. Flat Name Space

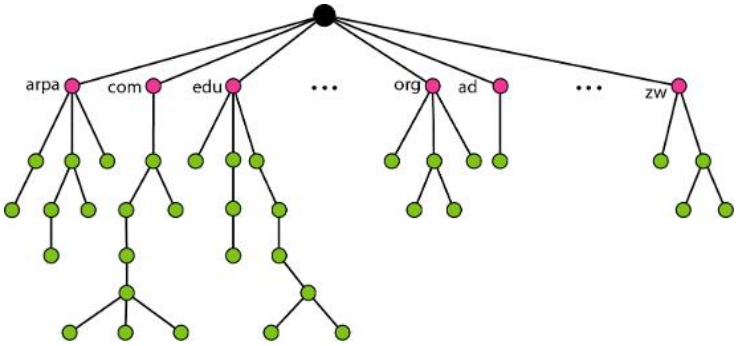
In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a fiat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

2. Hierarchical Name Space

In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, and the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility of the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources.

5. .2 DOMAIN NAME SPACE

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

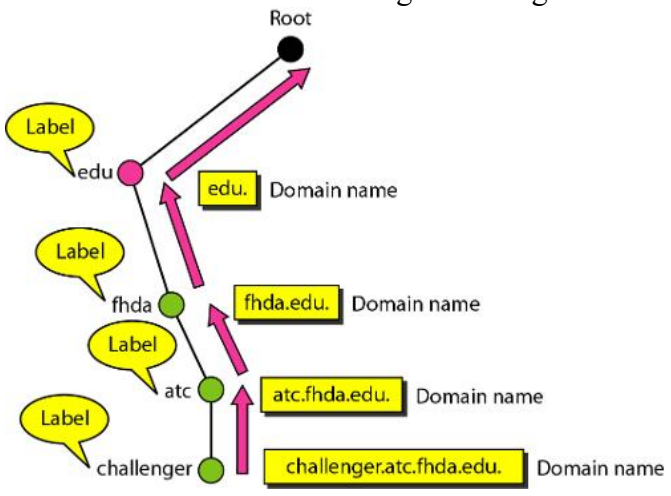


Label

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

Domain Name

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



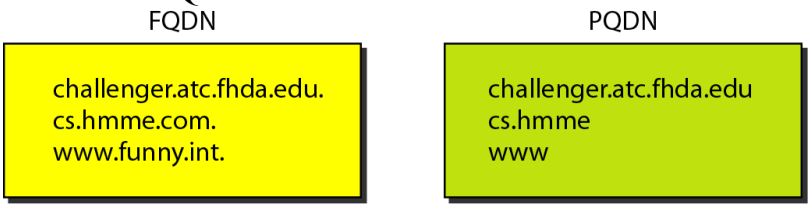
m ***m***

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host.

m ***m***

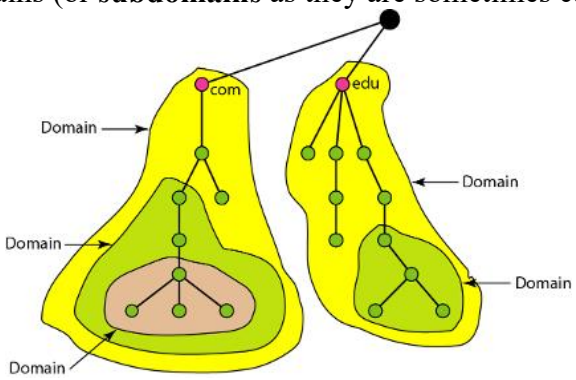
If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to

be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN.



Domain

A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Figure 25.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).

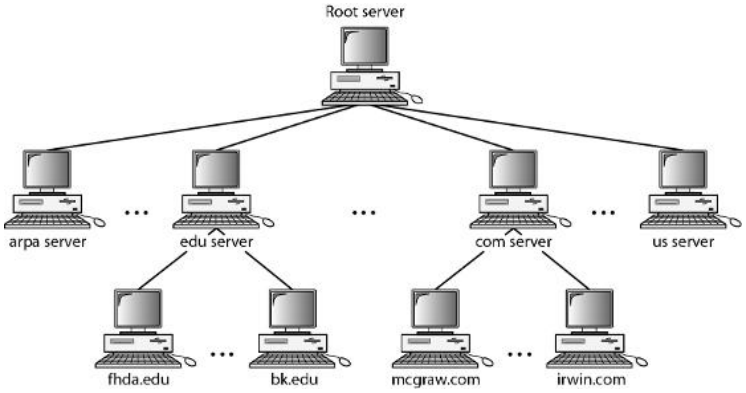


5. .3 DISTRIBUTION OF NAME SPACE

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. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not unreliable because any failure makes the data inaccessible.

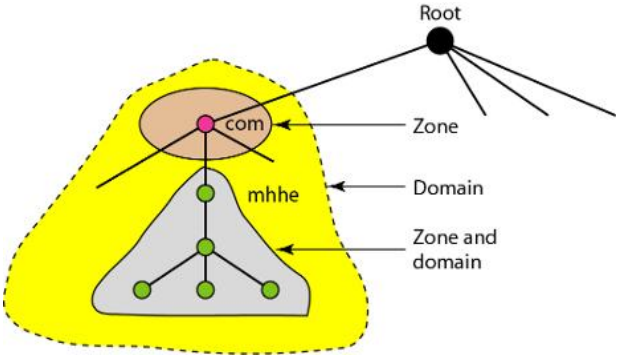
Hierarchy of Name Servers

The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or a small domain.



one

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



Root Server

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

Primary and Secondary Servers

DNS defines two types of servers: primary and secondary.

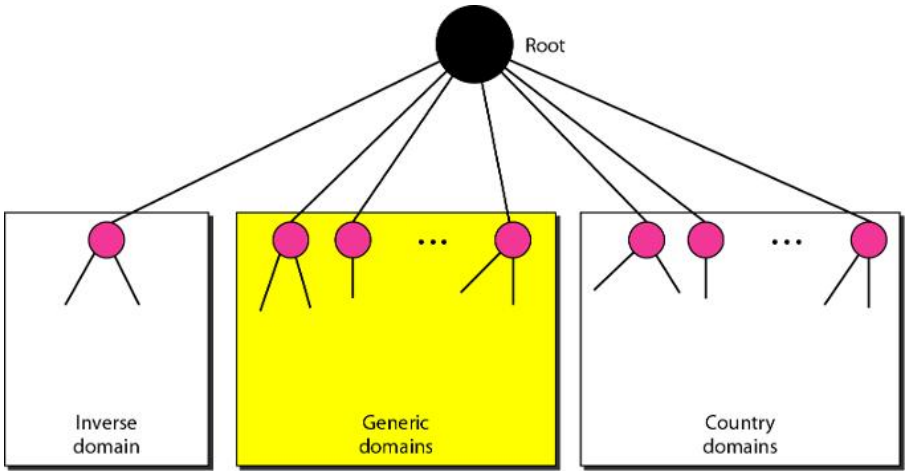
A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.

The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients.

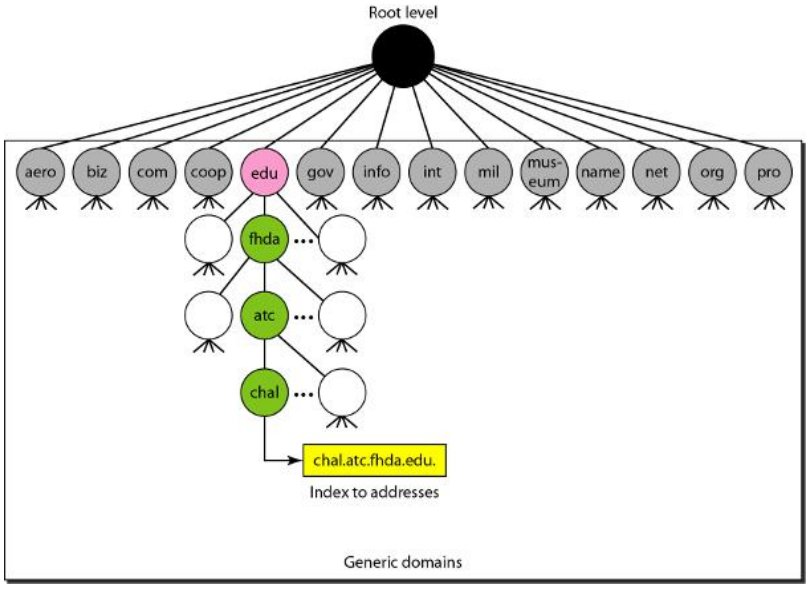
5. .4 DNS IN THE INTERNET

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



Generic Domains

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



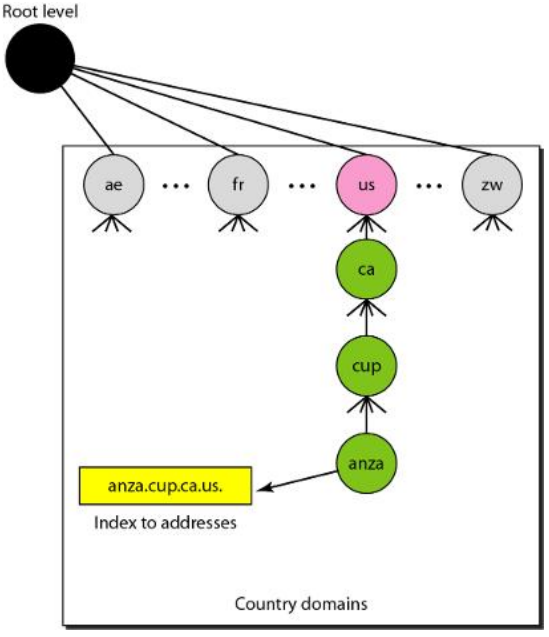
Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table

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

Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).

Below Figure shows the country domains section. The address *an a.cup.ca.us* can be translated to De Anza College in Cupertino, California, in the United States.

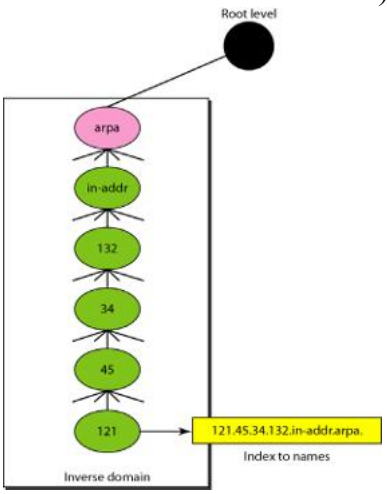


Inverse Domain

The inverse domain is used to map an address to a name. The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list.

This type of query is called an inverse or pointer (PTR) query. To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called *arpa* (for historical reasons). The second level is also one single node named *in-addr* (for inverse address). The rest of the domain defines IP addresses.

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



5. .5 RESOLUTION

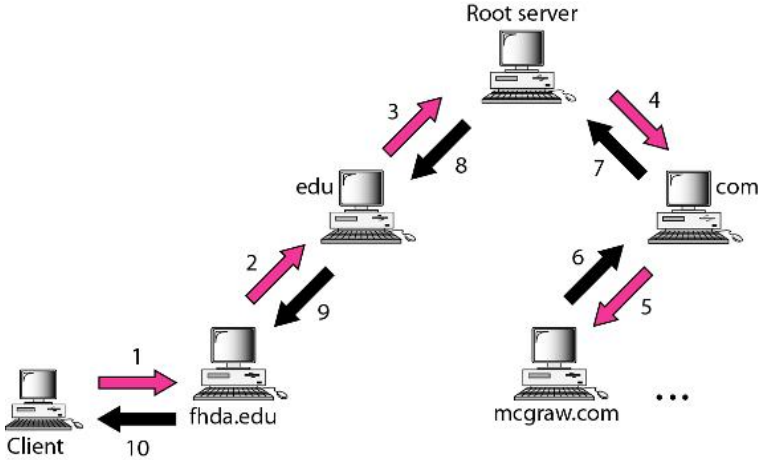
Mapping a name to an address or an address to a name is called *name-address resolution.*

Resolver

DNS is designed as a client/server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

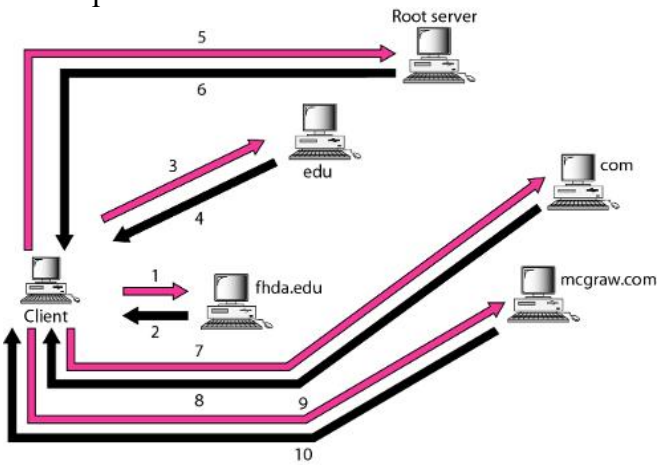
Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution.



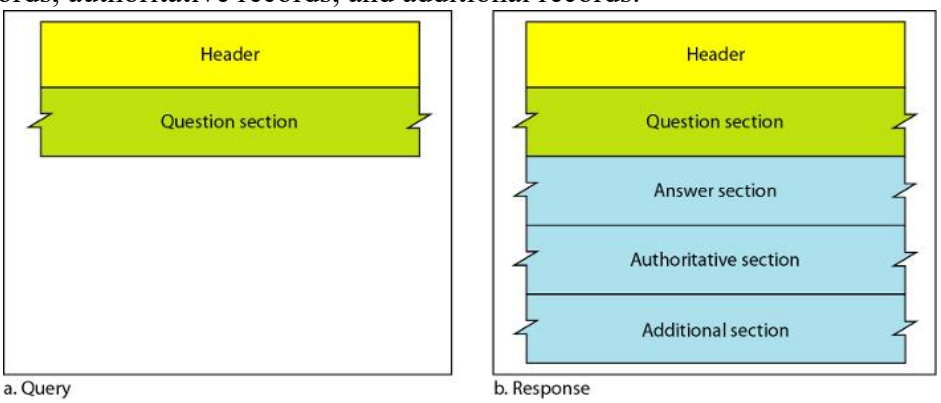
Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Now the client must repeat the query to the third server. This process is called iterative resolution.



5. . DNS MESSAGES

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



Header

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

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

The *identification* subfield is used by the client to match the response with the query. The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response. The *flags* subfield is a collection of subfields that define the type of the message, the type of answer requested, the type of desired resolution (recursive or iterative), and so on. The *number of question records* subfield contains the number of queries in the question section of the message. The *number of answer records* subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message. The *number of authoritative records* subfield contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message. Finally, the *number of additional records* subfield contains the number additional records in the additional section of a response message. Its value is zero in the query message.

This is a section consisting of one or more question records. It is present on both query and response messages. We will discuss the question records in a following section.

This is a section consisting of one or more resource records. It is present only on response messages. This section includes the answer from the server to the client (resolver).

This is a section consisting of one or more resource records. It is present only on response messages. This section gives information (domain name) about one or more authoritative servers for the query.

m

This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional information that may help the resolver.

5. . TYPES OF RECORDS

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

uestion Record

A question record is used by the client to get information from a server. This contains the domain name.

Resource Record

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

5. . REGISTRARS

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

http://www.intenic.net

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

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