MODULE 1

Introduction to networks

1.2 Network Hardware

NETWORKS: A network is a set of devices (often referred to as nodes) connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

Computer network: is a group of computers linked to each other that enables the computer to communicate with another computer and share their resources, data, and applications.

Network Hardware: is a set of physical or network devices that are essential for interaction and communication between hardware units operational on a computer network.
Ex: HUB, Router, Switch

Types of the Transmission Technologies:

Point-to-point Link	Broadcast Link
It is a communication over a link between single transmitter and a receiver.	It is a communication where single transmitter is linked to multiple receivers.
Only one receiver is involved.	Large num of receivers are involved.
Ex:Telephone	Ex:Radio
Point-to-point transmission with exactly one sender and exactly one receiver is sometimes called unicasting.	Broadcast systems also support transmission to a subset of the machines, which known as multicasting.

Unicast	Broadcast	Multicast
A communication where a message is sent from one sender to one receiver.	A communication where a message is sent from one sender to all receivers.	A communication where a message is sent from one sender to a group of receivers
Ex:Telephone	Ex:Radio	Ex:Video streaming, online gaming

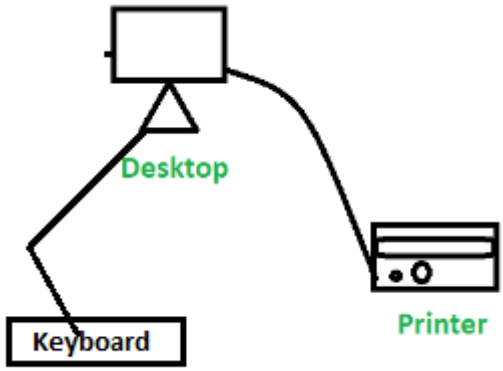
Types of Networks (Based on Size):

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

Personal Area Network: is the computer network that connects computers/devices within the range of an individual person.

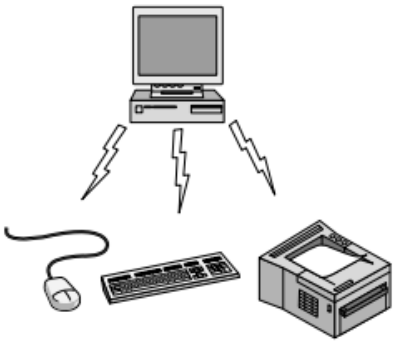
Wired PAN –

Wired PAN is connected through cables/wires.



Wireless PAN –

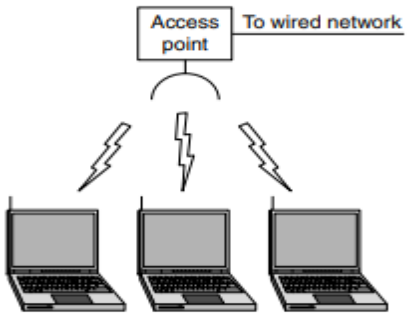
Wireless Personal Area Network (WPAN) is connected through signals such as infrared, ZigBee, Bluetooth etc.



Advantages	Disadvantages
• PAN is relatively flexible and provides high efficiency for short network ranges.	• Low network coverage area/range.
• It needs easy setup and relatively low cost.	• Limited to relatively low data rates.
• It is easy and portable.	• May not be suitable for large-scale data transfer or communication.

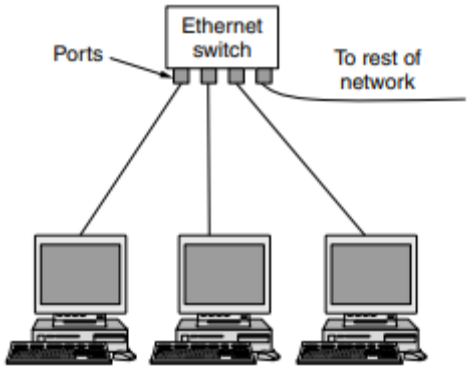
Local Area Network: A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. When LANs are used by companies, they are called **enterprise networks.** Data transfer rate is **10 to 100 mbps.**

Wireless LANs: Wireless LANs use high-frequency radio waves instead of cables for communications. They have an Access Point or a wireless router or a base station for transferring packets to and from the wireless computers and the internet. Most WLANs are based on the standard IEEE 802.11 or WiFi.



It runs at speeds anywhere from 11 to hundreds of Mbps.

Wired LANs: Wired LANs use a range of different transmission technologies. Most of them use copper wires, but some use optical fiber. Wired LANs run at speeds of 100 Mbps to 1 Gbps. IEEE 802.3, popularly called Ethernet, is, by far, the most common type of wired LAN.



Switched Ethernet is a modern version of the original Ethernet design that broadcast all the packets over a single linear cable It used a simple algorithm: computers could transmit whenever the cable was idle. If two or more packets collided, each computer just waited a random time and tried later. We will call that version **classic Ethernet.**

Virtual LAN or VLAN:

It is also possible to divide one large physical LAN into two smaller logical LANs.

Advantages	Disadvantages
• Faster Communication: High data transfer rate.	• Limited Range – One of the main disadvantages of LANs is that they have a limited range. This means that they can only cover a small area, such as a single building or group of buildings.
• Sharing Resources: A LAN enables users to share resources such as printers, scanners, and other hardware devices. Instead of having to buy a printer for every computer	• **Security Risks** – LANs can be vulnerable to security risks, such as hacking and unauthorized access. This is because LANs are often connected to the internet, which means that they can be accessed by anyone with the right skills and tools.
• Enhanced Security: LANs allow for enhanced security measures to be put in place, such as firewalls and encryption, to protect against unauthorized access to the network	• Maintenance and Upkeep – Another disadvantage of LANs is that they require regular maintenance and upkeep. This includes things like updating software, monitoring network traffic, and replacing faulty hardware.
• Centralized Management: A LAN enables	• **Cost** – Setting up and maintaining a LAN can be

centralized management of network devices, such as routers, switches, and servers.	expensive, especially if you need to purchase specialized hardware and software.

Metropolitan Area Network: Design to extend over a large area. Connecting number of LAN's to form larger network, so that resources can be shared. Networks can be up to 5 to 50 km. Owned by organization or individual. Data transfer rate is low compare to LAN. Example: Organization with different branches located in the city.

IEEE 802.16 and is popularly known as WiMAX.

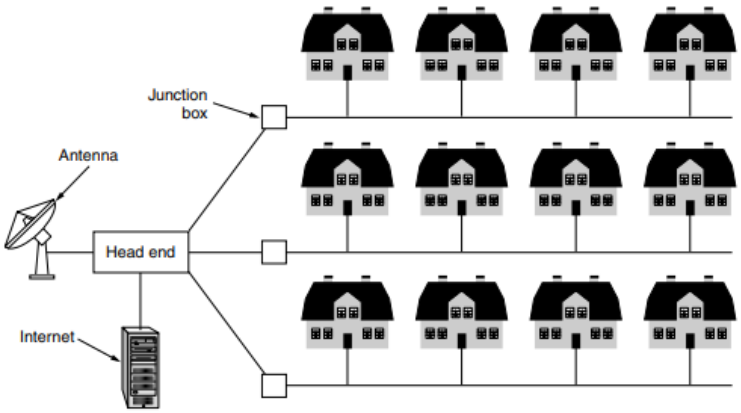


Figure 1-9. A metropolitan area network based on cable TV.

Advantages	Disadvantages
• **High Bandwidth Capacity** –which means it can transfer a lot of data quickly and efficiently.	• **High Cost** – Setting up a MAN can be a very expensive process.
• **Cost-effective** – A MAN is typically less expensive than a WAN and can be a cost-effective solution for organizations that need to connect their offices and buildings in a specific location.	• **Security Risks** – Because MANs are larger and more complex than Local Area Networks (LANs), they are also more vulnerable to security breaches.
• **Secure** – A MAN is a secure network that can protect your data from unauthorized access. This is especially important for organizations that deal with sensitive information.	• **Limited Access** – Because MANs are designed to cover a specific geographic area, they can be limiting in terms of access. If you need to access resources outside of this area, you may need to set up additional networks or rely on other solutions.
• **Scalable –** A MAN is scalable, which means it can grow with your organization.	

Wide Area Network: A WAN (Wide Area Network) spans a large geographical area, often a country or continent.

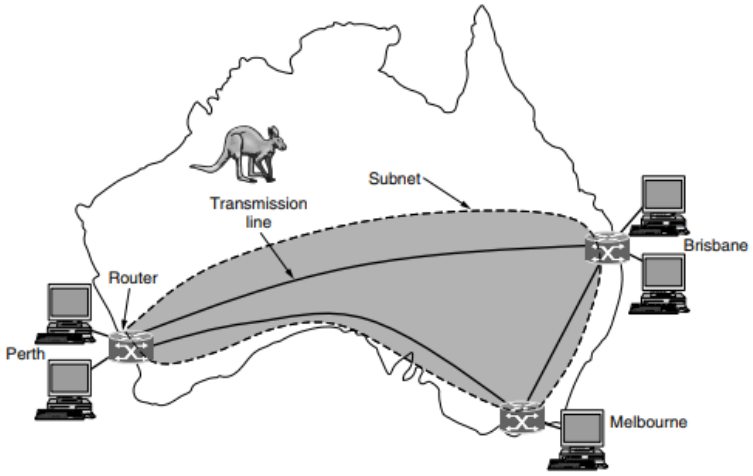


Figure 1-10. WAN that connects three branch offices in Australia.

VPN (Virtual Private Network):

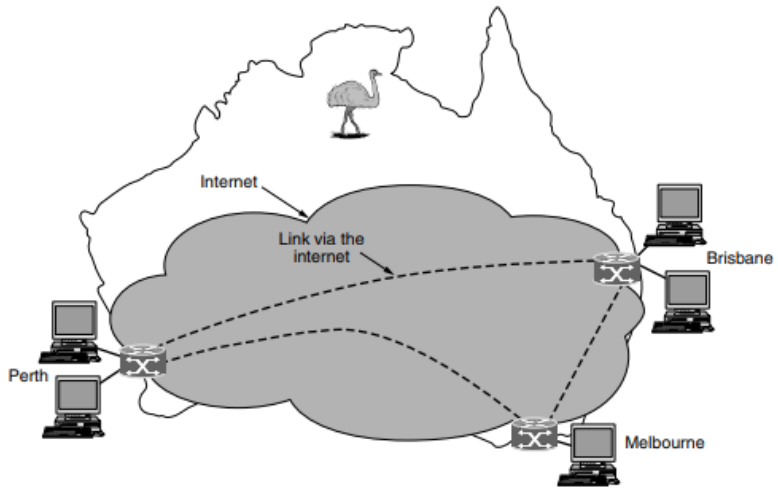


Figure 1-11. WAN using a virtual private network.

ISP network:

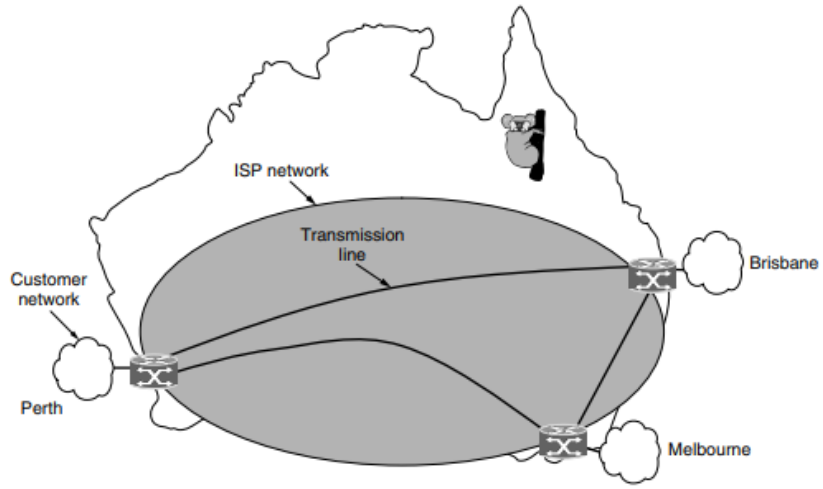


Figure 1-12. WAN using an ISP network.

Internetworks: A collection of interconnected networks is called an internetwork or internet. The Internet uses ISP networks to provide internet facility to the connected enterprise networks, home networks, and many other networks.

Subnet: collection of routers and communication lines owned by the network operator.

Ex: telephone system consists of telephone switching offices connected to one another by high-speed lines, and to houses and businesses by low-speed lines. These lines and equipment, owned and managed by the telephone company, form the subnet of the telephone system. The telephones themselves (the hosts in this analogy) are not part of the subnet.

A **network** is formed by the combination of a subnet and its hosts.

The general name for a machine that makes a connection between two or more networks and provides the necessary translation, both in terms of hardware and software, is a **gateway**.

1.3 Network Software

Network software refers to a set of programs and applications that enable communication, data transfer, and management of **computer networks**.

Protocol Hierarchies:

Protocol is an agreement between the communicating parties on how communication is to proceed. All protocols might be implemented using hardware, software, or combination of both of them.

Protocol Hierarchies :Networks are organized and arranged as a stack of layers of hardware and software, one on top of another.

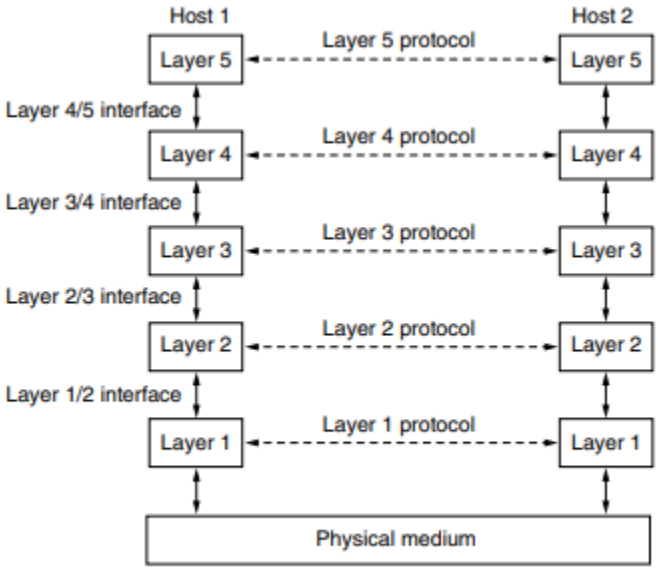


Figure 1-13. Layers, protocols, and interfaces.

- A set of layers and protocols is called **network architecture**.
- One protocol per layer is called a **protocol stack**.
- The entities comprising the corresponding layers on different machines are called **peers**.

- The peers may be software processes, hardware devices, or even human beings. In other words, it is the peers that communicate by using the protocol to talk to each other.
- Between each pair of adjacent layers is an **interface**. The interface defines which primitive operations and services the lower layer makes available to the upper one.

Example:

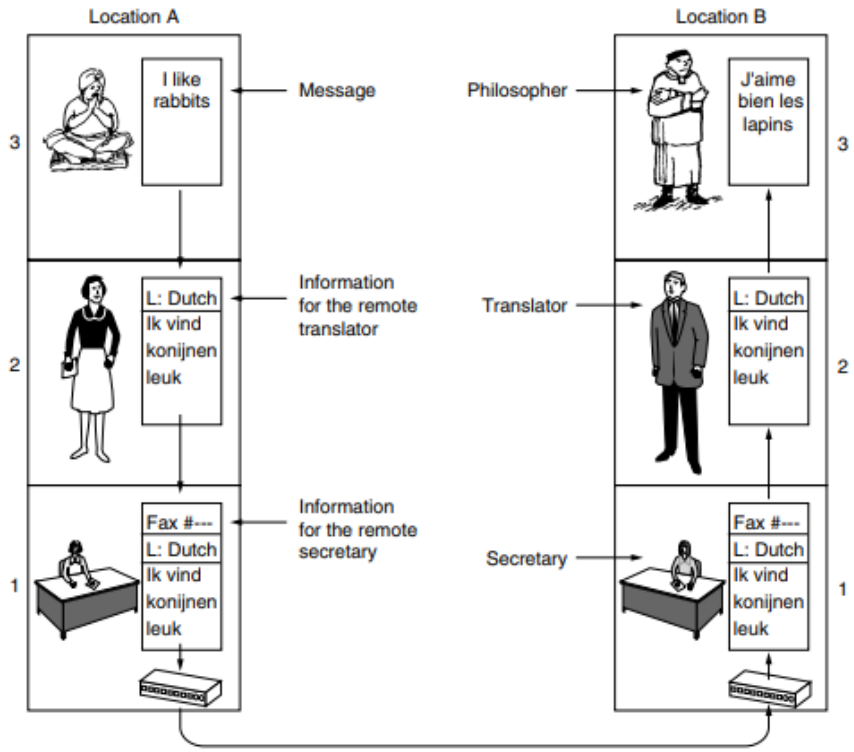


Figure 1-14. The philosopher-translator-secretary architecture.

Ex: Information Flow

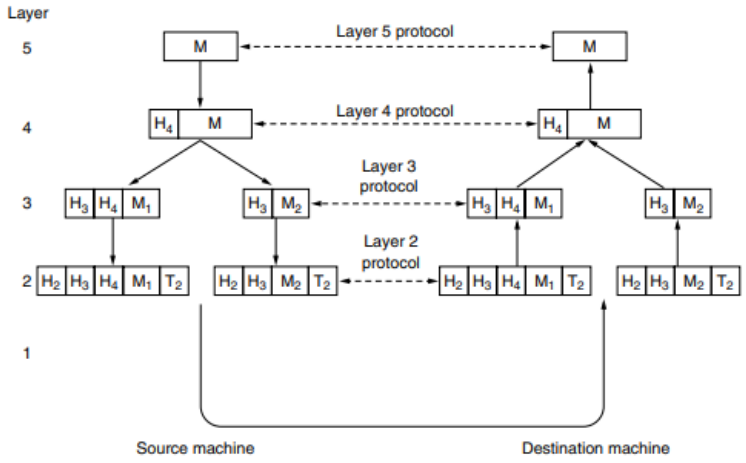


Figure 1-15. Example information flow supporting virtual communication in layer 5.

Sender Side:

- A message, M, is produced by an application process running in layer 5 and given to layer 4 for transmission.
- Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3. The header includes control information, such as addresses, sequence numbers (in case the lower layer does not preserve message order), sizes, and times.
- In many networks, no limit is placed on the size of messages transmitted in the layer 4 protocol but there is nearly always a limit imposed by the layer 3 protocol.
- Layer 3 must break up the incoming messages into smaller units, packets, prepending a layer 3 header to each packet.
- In this example, M is split into two parts, M1 and M2 that will be transmitted separately. Layer 3 decides which of the outgoing lines to use and passes the packets to layer 2.
- Layer 2 adds to each piece not only a header but also a trailer, and gives the resulting unit to layer 1 for physical transmission.

Receiver Side: The message moves upward, from layer to layer, with headers being stripped off as it progresses.

Design Issues for the Layers:

- **Reliability:** Network channels and components may be unreliable, resulting in loss of bits while data transfer. So, an important design issue is to make sure that the information transferred is not distorted.
  Solution: Error Detection and Correction
- **Routing:**
  The network layer chooses the most relevant and best path for the data transmission from source to destination and in a large network, there may be some links or routers that are broken. So, an important design issue is to make sure that the information is transferred to correct destination.
- **Addressing:** Maintains the address at the frame header of both source and destination and performs addressing to detect various devices in network.
- **Inter-networking:** not all communication channels preserve the order of messages sent on them, leading to solutions that number messages. Another example is differences in the maximum size of a message that the networks can transmit. This leads to mechanisms for disassembling, transmitting, and then reassembling messages.
- **Scalable:** Designs issue is to make sure that continue to work well when the network gets large.
- **Resource allocation:** Designs issue is to make sure those resources of one host does not interfere with another.
  Solution: Statistical multiplexing is a time-division multiplexing technique in which time slots are dynamically allocated on the basis of need.

  Congestion: Network is oversubscribed because too many computers want to send too much traffic, and the network cannot deliver it all.
  Flow control: It is a set of procedures that tells the sender how much data it can transmit before the data overwhelms the receiver.

Quality of service: Is the use of mechanisms or technologies that work on a network to control traffic and ensure the performance of critical applications with limited network capacity.

- **Security**: Designs issue is to make sure that there is no eavesdropping on communications
  Solution: confidentiality, authentication and integrity

Connection Oriented VS Connectionless Services:

Connection Oriented	Connectionless
Service is related to the telephone system.	Service is related to the postal system.
It includes connection establishment and connection termination.	It does not include any connection establishment and connection termination
In connection-oriented Service, Packets follow the same route.	In connection-less Service, Packets do not follow the same route.
No congestion occurs due to an organized data transfer.	It is very likely to cause congestion in a network.
Transferring data in a connection-oriented service takes up a higher bandwidth.	A connectionless service requires a comparatively lower bandwidth for transferring the data.
Alternative name: Circuit Switching	Alternative name: store-and-forward switching, cut-through switching

Unreliable (meaning not acknowledged) connectionless service is often called **datagram service**, in analogy with telegram service, which also does not return an acknowledgement to the sender.
The **acknowledged datagram** service: Text messaging on mobile phones is an example.
Request-reply service: the sender transmits a single datagram containing a request; the reply contains the answer.

Different types of Services

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Movie download
	Unreliable connection	Voice over IP
Connectionless	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Text messaging
	Request-reply	Database query

Figure 1-16. Six different types of service.

Reliable byte stream in which the bytes which emerge from the communication channel at the recipient are exactly the same, and in exactly the same order, as they were when the sender inserted them into the channel.
(message is one contiguous stream of bytes.,Junk mail()spam)).

Service Primitives: A service is a set of primitives or we call it as operations where a user can invoke to access the service.

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Figure 1-17. Six service primitives that provide a simple connection-oriented service.

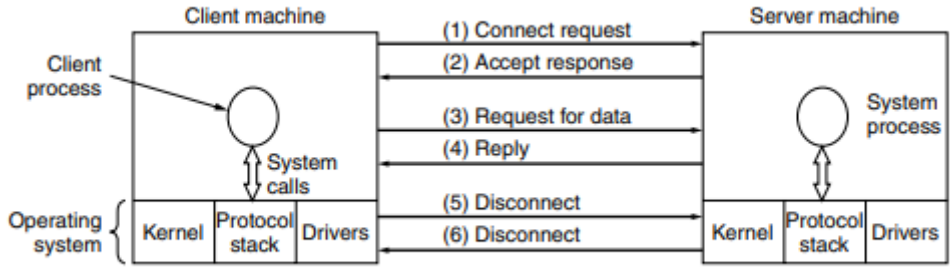


Figure 1-18. A simple client-server interaction using acknowledged datagrams.

Following are the Steps to be executed:

- First, the **server** executes **LISTEN** to indicate that it is prepared to accept in-coming connections.
  After executing the primitive, the server process is blocked until a request for connection appears.
  A common way to implement LISTEN is to make it a blocking system call.
- Next, the **client** process executes **CONNECT** to establish a connection with the server.
  The client process is suspended until there is a response.
  The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server's address.

1. The client machine sends the Connect request to the server machine to establish a connection by specifying the server's address.
2. When request packet arrives at the server, it checks for listener and unblocks it. Server sends a response back to the client process to accept the connection request with ACCEPT call. The arrival of this response then unblocks the client.
3. Now, server will execute RECEIVE to prepare to accept the first request. The RECEIVE call blocks the server. Then the client executes SEND to transmit its request followed by the execution of RECEIVE to get the reply.
4. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the server uses SEND to return the answer to the client.
5. When the client is done, it executes DISCONNECT to terminate the connection.
6. When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection.

The Relationship of Services to Protocols:

A **service** is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented.

A **protocol**, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users.

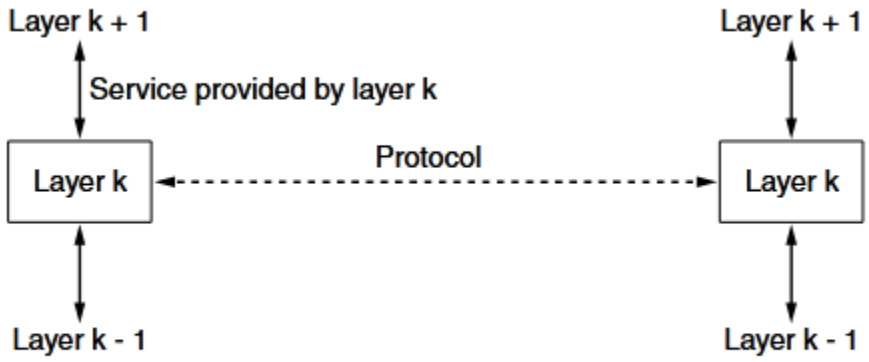


Figure 1-19. The relationship between a service and a protocol.

1.4 Reference Models

Two Types:

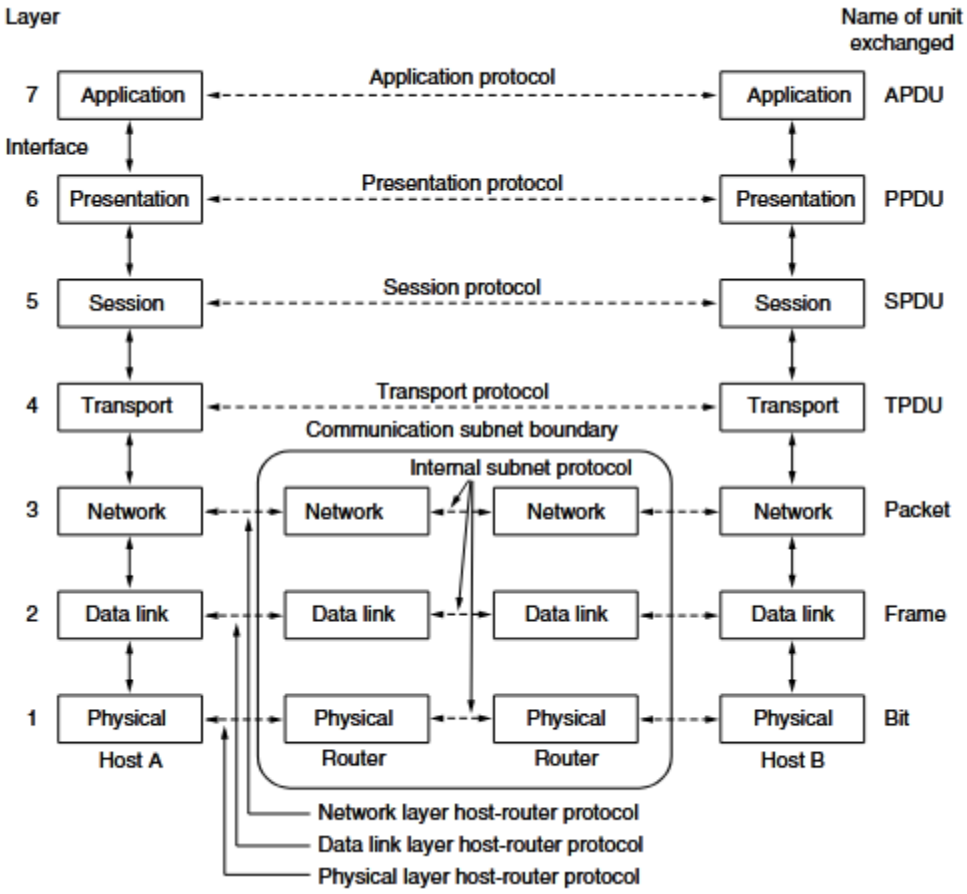
- The OSI (Open Systems Interconnection) Reference Model
- The TCP/IP Reference Model

OSI (Open Systems Interconnection)

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

The Principles:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.
5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.



1. **The Physical Layer:**

- The lowest layer of the OSI reference model is the physical layer.
- The physical layer contains information in the form of **bits.** It is responsible for transmitting individual bits from one node to the next.

Design Issues:

- Make sure that when one side sends a 1 bit it is received by the other side as a 1 bit, not as a 0 bit.
- Make sure that what electrical signals should be used to represent a 1 and a 0.
- How many nanoseconds a bit lasts?
- Whether transmission may proceed simultaneously in both directions?
- How the initial connection is established and terminated?

2. **The Data Link Layer:**

- The data link layer is responsible for the node-to-node delivery of the message.
- The main function of this layer is to make sure data transfer is error-free from one node to another, over the physical layer.

- The sender breaks up the input data into data frames and transmits the frames sequentially.

Design Issues:

- Make sure that receiver receives a correct frame and send back the acknowledged frame if service is reliable.
- How to keep a fast transmitter from drowning a slow receiver in data(Flow Control).
- How to control access to the shared channel?(medium access control)

3. The Network Layer:

- The network layer works for the transmission of data from one host to the other located in different networks.
- The network layer controls the operation of the subnet.

Design Issues:

- Determining how packets are routed from source to destination. Routes can be based on static tables that are ‘‘wired into’’ the network and rarely changed, or more often they can be updated automatically to avoid failed components.
- How to handle the congestion? (If too many packets are present in the subnet at the same time, it leads to congestion).
- How to handle the problems to allow heterogeneous net-works to be interconnected. (The addressing used by the second network may be different from that used by the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on).

4. The Transport Layer:

- The transport layer provides services to the application layer and takes services from the network layer.
- It is responsible for the End to End Delivery of the complete message.
- The transport layer also provides the acknowledgment of the successful data transmission and re-transmits the data if an error is found.

Design Issues:

- Accepting data from Session layer, split it into segments and send to the network layer.
- Ensure correct delivery of data with efficiency.
- Isolate upper layers from the technological changes.
- Error control and flow control.

5. The Session Layer:

The session layer allows users on different machines to establish sessions between them.

Services:

- Dialog control (keeping track of whose turn it is to transmit).
- Token management (preventing two parties from attempting the same critical operation simultaneously.
- Synchronization (check pointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery)

6. The Presentation Layer:

- Presentation layer is concerned with the syntax and semantics of the information transmitted.
- The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records) to be defined and exchanged.
- Encoding and decoding takes place.

7.The Application Layer:

- It receives information directly from users and displays incoming data to the user.
  Protocols:
- **HTTP (Hyper Text Transfer Protocol):** This is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server hosting the page using HTTP. The server then sends the page back.
- Other application protocols are used for file transfer(**FTP**), electronic mail(**SMTP**), and network news(**NNTP**).

The TCP/IP Reference Model

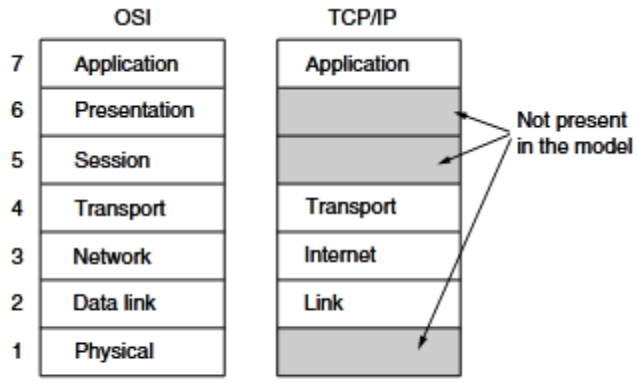


Figure 1-21. The TCP/IP reference model.

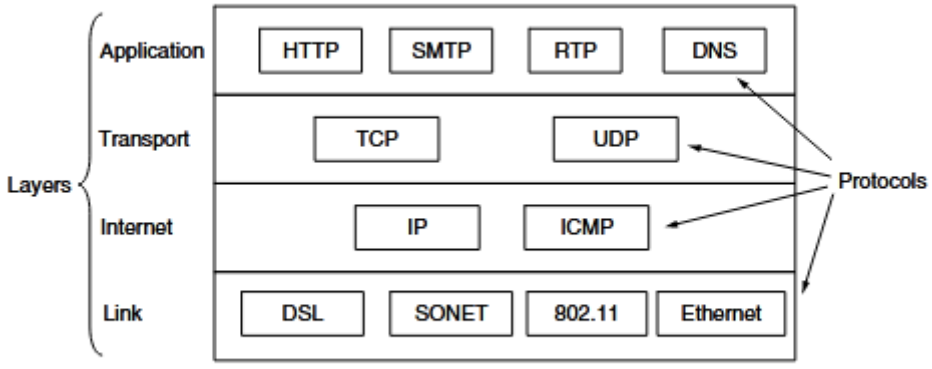


Figure 1-22. The TCP/IP model with some protocols we will study.

The Link Layer:

The lowest layer in the model, the link layer describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer.

The Internet Layer:

Protocols:

IP(Internet Protocol): It is a connectionless and unreliable protocol that provides a best effort delivery service.

ICMP (Internet Control Message Protocol): It monitors sending the queries as well as the error messages.

IGMP (Internet Group Message Protocol) –It allows the transmission of a message to a group of recipients simultaneously.

The Transport Layer:

TCP(Transmission Control Protocol)	**UDP(User Datagram Protocol)**
TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
Supports: Error Control, Flow Control and Congestion Control.	Does not support Flow Control and Congestion Control and limited Error Control.
Used for: Sending Emails, Transferring Files.	Used for: Video Streaming,Online Video Chats
TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.

The Application Layer:

virtual terminal (**TELNET**), file transfer (**FTP**), and electronic mail (**SMTP**), Domain Name System (**DNS**), for mapping host names onto their network addresses, **HTTP**, the protocol for fetching pages on the World Wide Web,and **RTP**, the protocol for delivering real-time media such as voice or movies.

Module 1

The Physical Layer of the OSI Model

Guided Transmission Media

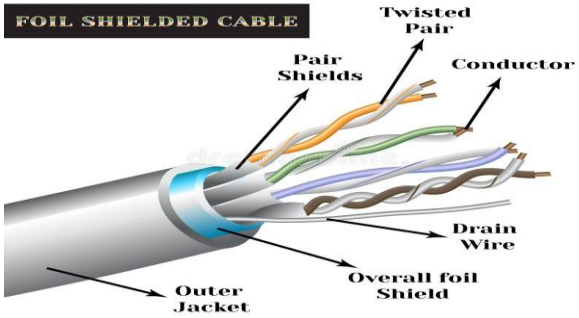
Guided Transmission Media

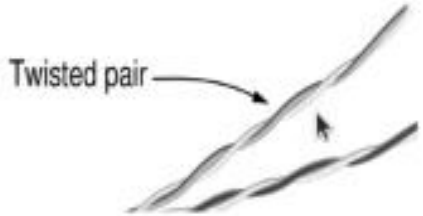
- Magnetic media
- Twisted pairs
- Coaxial cable
- Power lines
- Fiber optics

Magnetic Disc

- Write data onto magnetic media
  - Disks (e.g., DVD)
  - Tapes
- Data transmission speed
  - Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.
- Ultrium Tapes – 800 Gb Uncompressed Data
  - 60x60x60 cm can hold about 1000 tapes
  - Total Capacity 800 TeraBytes or 6400 terabits (6.4 peta bits)
  - A box of tapes can be delivered in the U.S. in 24 hours
  - The effective bandwidth is 6400 terabits/86,400 sec
  - or over 70 Gbps.
  - It destination is only one hour away bandwidth is increased
  - to over 1700 Gbps.
  - No computer network can approach this rate.
  - 1000 Tapes x $40 = $4000
  - Shipping $1000, Total $5000 roughly for 800 TB => 0.5 cent per GB.

Twisted Pair





- A twisted pair consists of two insulated copper wires, typically about 1 mm thick.
- Twisted together in a helical form, just like a DNA molecule
- Why twisting ?
- 2 parallel wires constitute a fine antenna.
- When the wires are twisted, the waves from different twists cancel out, so the wire radiates less effectively.
- Signal xmsn → difference in voltage between the two wires → better immunity to external noise
- As noise tends to affect both wires the same, leaving the differential unchanged.

- Telephony Lines (ADSL) – Twisted pair of wires.
- Category 5
  - 100 Mbps Ethernet uses two pairs; one pair for each direction.
  - 1 Gbps Ethernet uses all four pairs in both directions simultaneously.
- Definitions:
  - Full-duplex link
  - Half-duplex link
  - Simplex link
- Cat 6 - UTP - only wires and insulators.
- Cat 7 – STP - shielding on tps and around the entire cable - reduces the external interference and cross-talk
- Cat 6, Cat 7 needed to carry rates of 10 Gbps and higher.

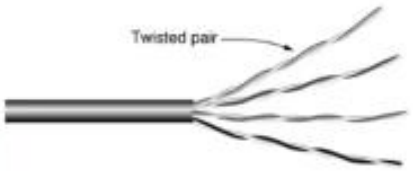
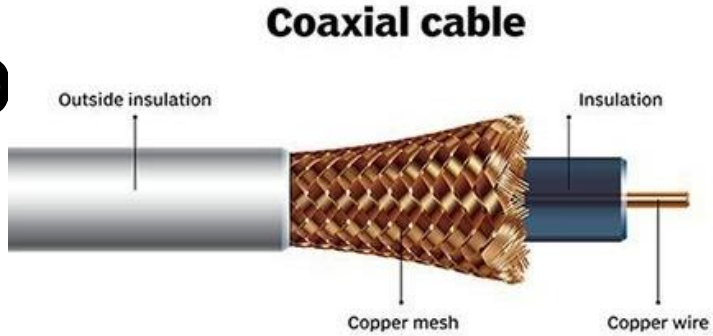


Fig. Category 5 UTP cable with four twisted pairs

Coaxial Cab



- Construction provides higher bandwidth and better noise immunity than UTP.
  - 50 ohm is used for digital transmission
  - 75 ohm is used for analog transmission and cable television.
- Bandwidth – few GHz
- Widely used for cable televisions and metropolitan are networks.

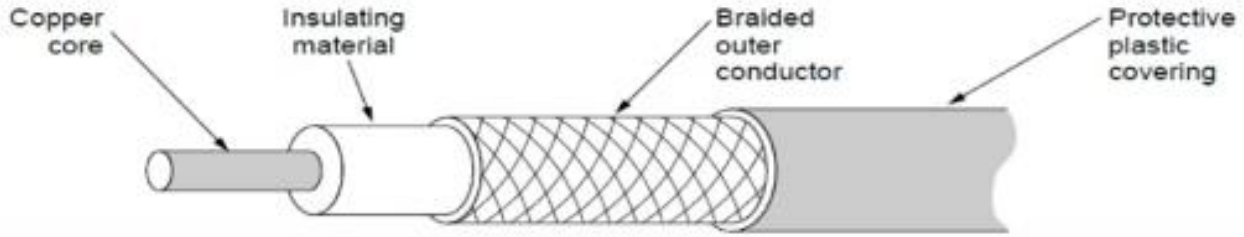


Fig. A coaxial cable

Power Lines

- High-rate communication of interest recently.
  - Inside home as a LAN
  - Outside home for broadband Internet access.
- Convenient way to perform networking
- Problem with wiring since they do attenuate significantly for higher frequencies.
- International standards are under development.

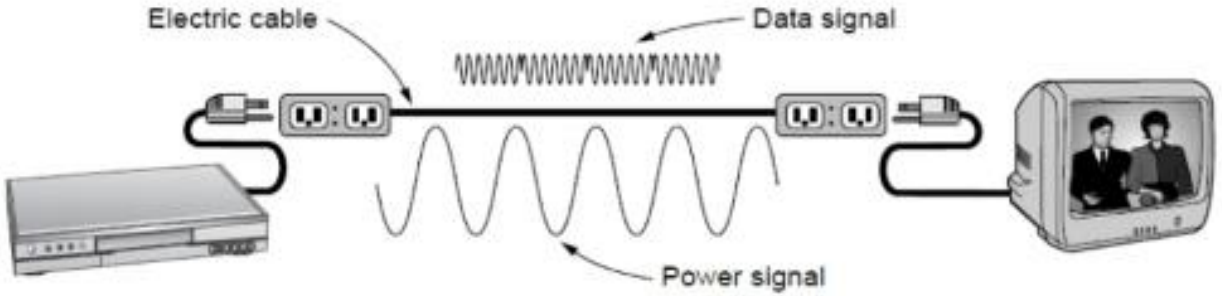
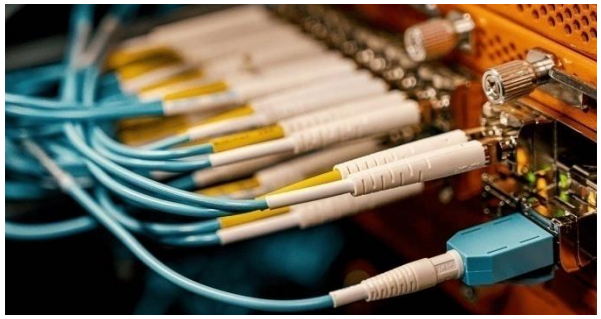


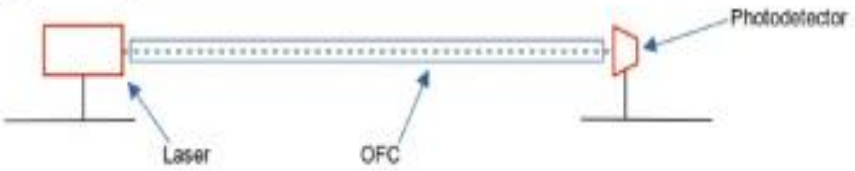
Fig. A network that uses household electrical wiring.

Fiber Optics



- Moore's Law – doubling of the number of transistors per chip roughly every 2 years:
  - Original IBM PC clock speed 4.77 MHz
  - 28 years latter PC's run four-core CPU's at 3 GHz. (Intel core i7 - 3.9 GHz in 2017)
  - Increase of a factor of around 2500, 16 per decade.
- Wide area communication links went
  - From 45 Mbps to 100 Gbps.
  - Achievable bandwidth with fiber technology is in excess of 50,000 Gbps (50 Tbps)
  - This 100 Gbps hard threshold is due to our inability to convert between electrical and optical signals any faster.
- Race between computing power and communication.

- By convention:
  - A pulse of light indicates a 1 bit, and
  - the absence of light indicates a 0 bit
- An optical transmission system has 3 key components :
  - The light source
  - The transmission medium, and
  - The detector.



- In this way, we have:
  - A unidirectional data transmission system
  - Accepts an electrical signal,
  - Converts and transmits it by light pulses, and
  - Reconverts back the output to an electrical signal at the receiving end.

- Receiving end of an optical fiber consists of a photodiode, which gives off an electrical pulse when struck by light.
- The response time of photodiodes that convert the signal from the optical to the electrical domain limits data rates to about 100 Gbps.
- Terminal noise is also an issue hence the light should be injected with sufficient energy in order to be detected.
- By making the pulses powerful enough the error rate can be made arbitrarily small.

An interesting principle of physic doesn't allow this system to leak light and thus be useful in practice

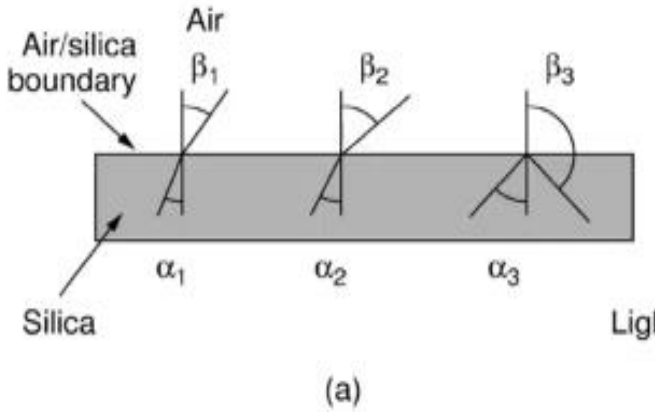


Fig. (a) 3 examples of a light ray from inside a silica fiber impinging on the air/silica boundary at different angles.

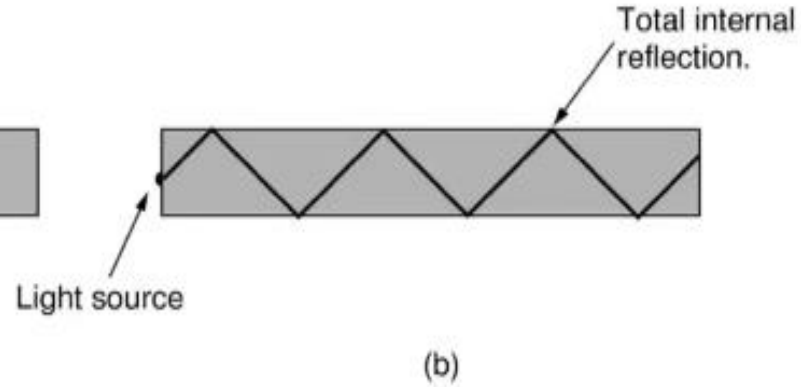


Fig. (b) Light trapped by total internal reflection.

- Every light ray that has incidence angle equal to or greater than the critical angel will be refracted internally: **multimode fiber.**

- If fiber's diameter is reduced to a few wavelength's of light it acts like a wave guide and the light can propagate only in a straight line without bouncing: **single-mode fiber**.

- Transmit data rate of 100 Gbps for 100 km without amplification.

- Attenuation of light through glass depends on the wavelength of the light (as well as on some physical properties of the glass).
- It is defined as the ratio of input to output signal power.
  - Figure on the next slide shows the attenuation in units of decibels per linear kilometer of fiber.

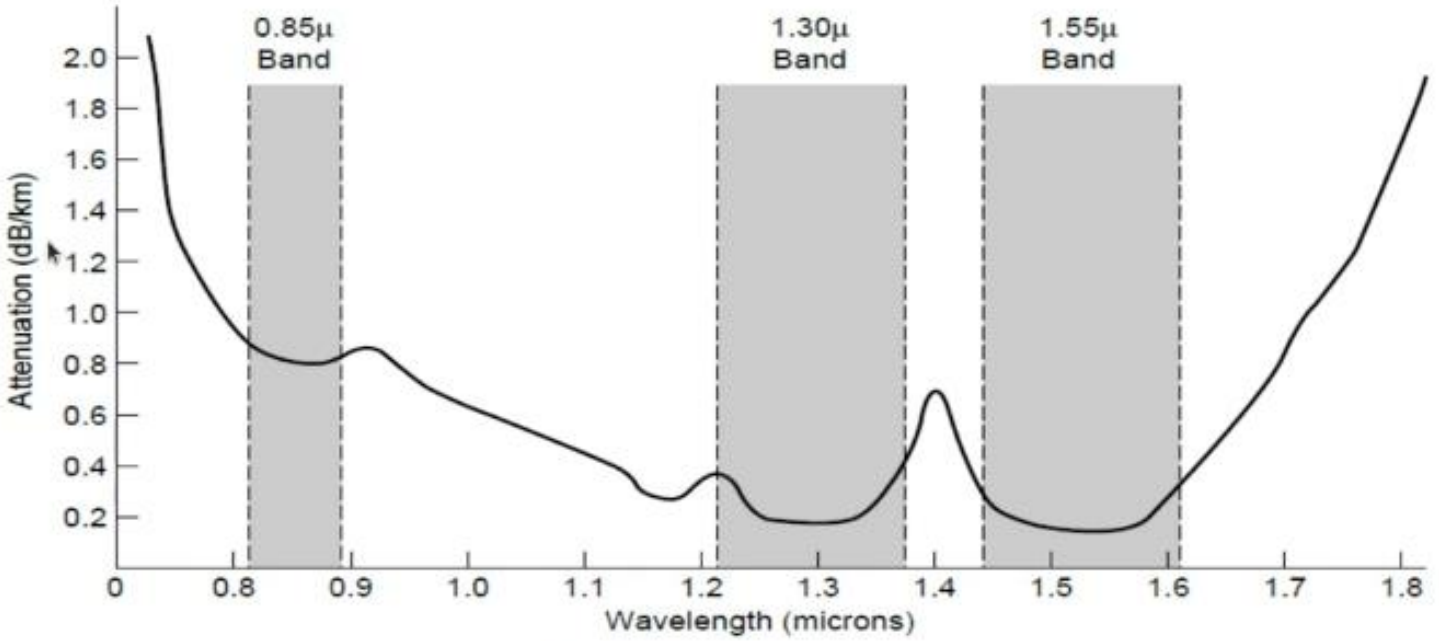


Fig. Attenuation of light through fiber in the infrared region

- Three wavelengths bands are most commonly used at present for optical communication:
  - 0.85μ
  - 1.30μ and
  - 1.55μ
  - They have bands ranging from 25,000 to 30,000 GHz wide.
- **Chromatic dispersion** – the spreading of wave as it propagates down on a fiber. To reduce, increase distance bet 2 signals → decreases signaling rate !
- **Solitons** – making the pulses in a special shape related to the reciprocal of the hyperbolic cosine – called solitons – cancels dispersion → possible to send pulses for thousands of kilometers without appreciable shape distortion. This was achieved in lab env.

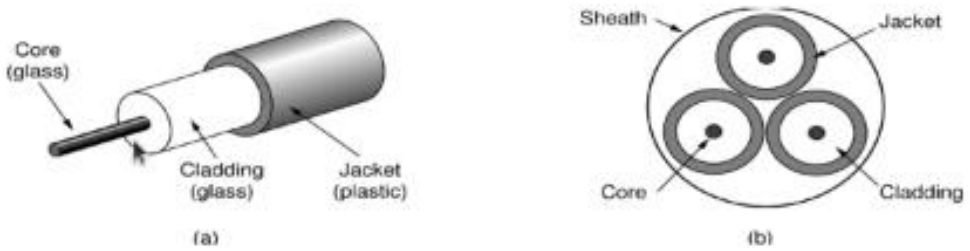


Fig. Views of a fiber cable

- Similar to Coaxial except they are without the braid.
- In Multimode fibers, core is typically 50 microns (~ thickness of human hair).
- In Single-mode fibers the core is 8-10 microns.
- Core is enclosed in by glass cladding with a lower index of refraction than the core to keep all the light in the core.
- Thin plastic jacket to protect the cladding.
- Fibers are typically grouped in bundles protected by an outer sheath.

- Placement of FOCs :
  - Terrestrial fiber sheaths - laid on the ground within a meter of the surface,
  - Near shore - transoceanic fiber cables are buried in trenches
  - Deep sea - laid on the see floor.
- Fibers are connected in three different ways:
  1. **Terminated in connectors** and are plugged into fiber sockets; connectors lose about 10%-20% of the light but they make it easy to reconfigure systems
  2. **Spliced** mechanically by carefully cutting ends next to each other and placing within a special sleeve and clamping them in place. Alignment can be improved by passing light through the junction and then making adjustments as necessary.
  3. Two pieces of fiber can be **fused** (melted) together to form a solid connection. Fused cable can be almost as good as a single drawn fiber. But here also is some amount of loss.
  4. For all three kinds of splices, reflections can occur at the point of the splice, and the reflected energy can interfere with the signal.

- Two types of technologies are used to drive the signals:
  - LED (Light Emitting Diodes), and
  - Semiconductor Lasers.
- They can be tuned in wavelengths by inserting bet src and fiber:
  - Febry-Perot interferometers, or
  - Mach-Zehnder interferometers
- **Febry-Perot interferometers** are simple resonant cavities consisting of two parallel mirrors. The light is incident perpendicular to the mirrors. The length of the cavity selects out those wavelengths that fit inside an integral number of times.
- **Mach-Zehnder interferometers** separate the light into two beams. The beams travel slightly different distances. Then they are recombined at the end and are in phase for only certain wavelengths.

Item	LED	Semiconductor laser
Data rate	Low	High
Fiber type	Multi-mode	Multi-mode or single-mode
Distance	Short	Long
Lifetime	Long life	Short life
Temperature sensitivity	Minor	Substantial
Cost	Low cost	Expensive

A comparison of semiconductor diodes and LEDs as light sources

- Fiber has numerous advantages:
  - Higher bandwidths,
  - Low attenuation (50 km vs. 5 km repeaters)
  - Not affected by power surges, electromagnetic interference
  - Not affected by corrosive chemicals in the air.
  - It is thin and lightweight → lower installation cost
  - Cable ducts are full – hence removing copper wires and replacing them with Fiber is good alternative.
  - Security against wiretappers : fibers do not leak light and are difficult to tap.

- Disadvantages of Fiber Optics:
  - Requires highly skilled workers,
  - They can be damaged easily,
  - Two way transmission requires either two different fibers or two frequency bands on one fiber.
  - Finally, fiber interfaces cost more than electrical interfaces.

Module 2
The Data link layer

The Data link layer:

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

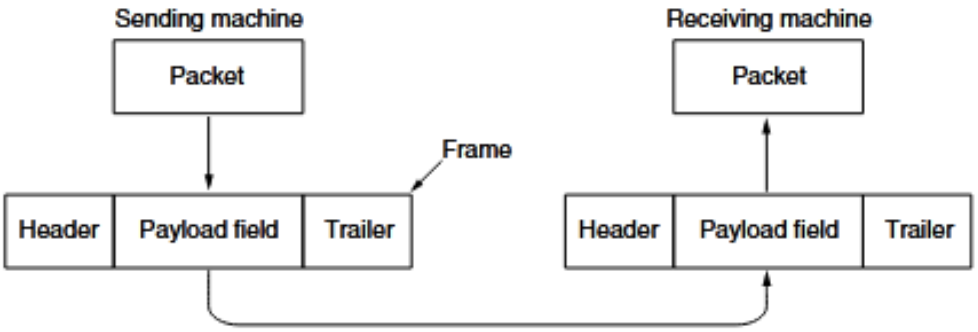


Figure 3-1. Relationship between packets and frames.

- 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.

3.1 Design issues of DLL:

1. **Services provided to the network layer –**

   The data link layer act as a service interface to the network layer. The principle service is transferring data from network layer on sending machine to the network layer on destination machine.

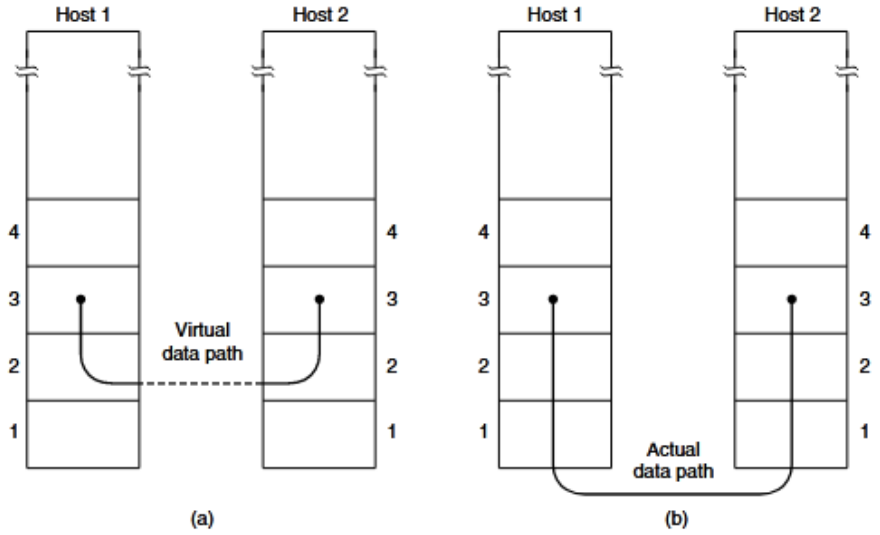


Figure 3-2. (a) Virtual communication. (b) Actual communication.

The types of services provided can be of three types −

- **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, no attempt is made to detect the loss or recover from it in the data link layer.
- **Acknowledged connectionless service**- hen 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.

  Drawbacks:
- **Acknowledged 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.

  If acknowledged connectionless service were used, it is conceivable that lost acknowledgements could cause a frame to be sent and received several times, wasting bandwidth.

  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.

2. **Framing –**

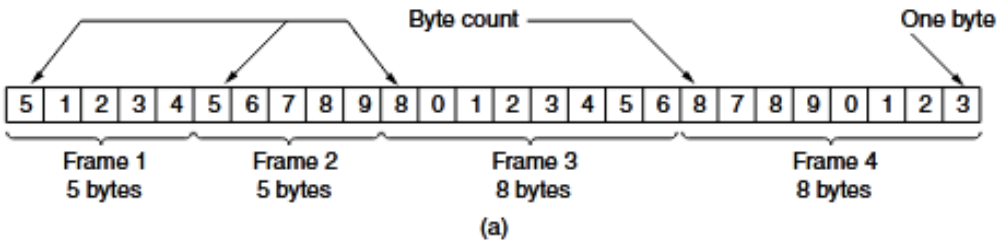
   The data link layer encapsulates each data packet from the network layer into frames that are then transmitted.

   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.

   Four Methods to perform framing:

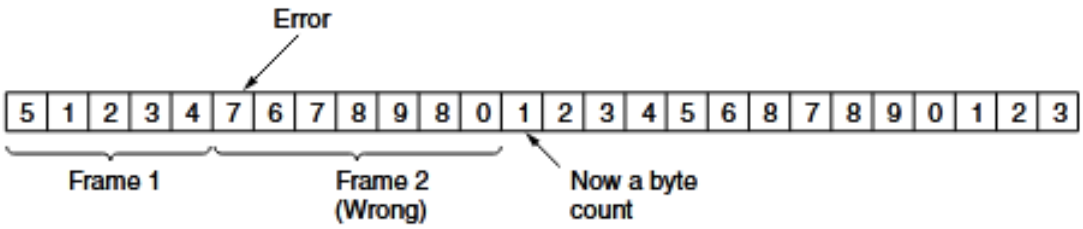
   1. Byte count.
   2. Flag bytes with byte stuffing.
   3. Flag bits with bit stuffing.
   4. Physical layer coding violations.

1. Byte count: The first 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.



(a)

Drawback:



(b)

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. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many bytes to skip over to get to the start of the retransmission. For this reason, the byte count method is rarely used by itself.

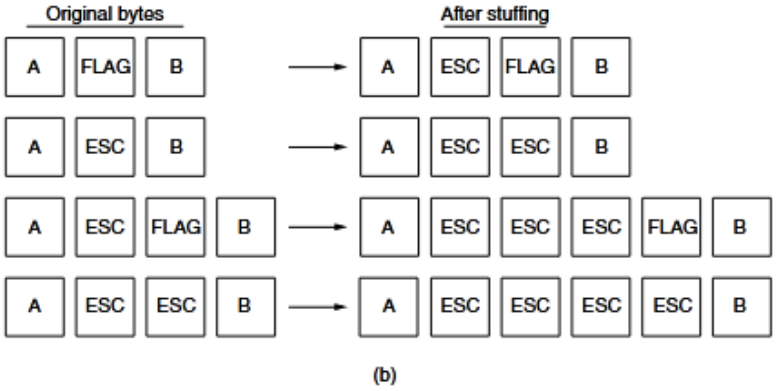
2. Flag bytes with byte stuffing: The second framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. Often the same byte, called a flag byte, is used as both the starting and ending delimiter. This byte is shown in Fig. (a) as 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.

(a)

Drawback:

What if flag byte occurs in the data?

Solution: byte stuffing



(b)

3. Flag bits with bit stuffing:

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

5. Bit stuffing. (a) The original data. (b) The data as they
(c) The data as they are stored in the receiver's memory af

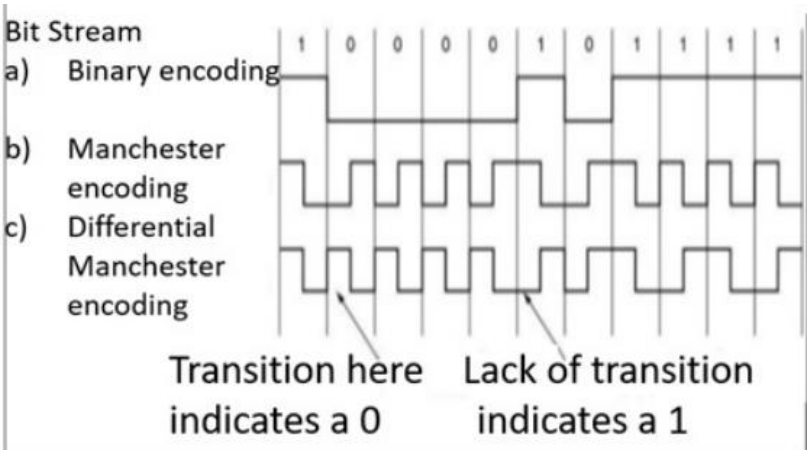
In Data Link layer, the stream of bits from the physical layer is divided into data frames. The data frames can be of fixed length or variable length. In variable - length framing, the size of each frame to be transmitted may be different. So, a pattern of bits(01111110) is used as a delimiter to mark the end of one frame and the beginning of the next frame.

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

Drawback:

With both bit and byte stuffing, a side effect is that the length of a frame now depends on the contents of the data it carries. For instance, if there are no flag bytes in the data, 100 bytes might be carried in a frame of roughly 100 bytes. If, however, the data consists solely of flag bytes, each flag byte will be escaped and the frame will become roughly 200 bytes long.

4. Physical layer coding violations: This framing method is used only in those networks in which encoding on the physical medium contain some redundancy. Some LANs encode each bit of data by using two physical bits that Manchester coding uses.Here, Bit 1 is encoded into a high-low (10) pair and Bit 0 is encoded into a low-high (01) pair.



3. Error Control –

Error control is done to prevent duplication of frames. The errors introduced during transmission from source to destination machines must be detected and corrected at the destination machine.

Possible ways:

If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely. On the other hand, a negative acknowledgement means that something has gone wrong and the frame must be transmitted again. If ACK is lost…

- This possibility is dealt with by introducing timers into the data link layer. When the sender transmits a frame, it generally also starts a timer. The timer is set to expire after an interval long enough for the frame to reach the destination, be processed there, and have the acknowledgement propagate back to the sender. Normally, the frame will be correctly received and the acknowledgement will get back before the timer runs out, in which case the timer will be canceled.
- However, if either the frame or the acknowledgement is lost, the timer will go off, alerting the sender to a potential problem. The obvious solution is to just transmit the frame again.
- However, when frames may be transmitted multiple times there is a danger that the receiver will accept the same frame two or more times and pass it to the network layer more than once. To prevent this from happening, it is generally necessary to assign sequence numbers to outgoing frames, so that the receiver can distinguish retransmissions from originals.

4. Flow Control –

The data link layer regulates flow control so that a fast sender does not drown a slow receiver. When the sender sends frames at very high speeds, a slow receiver may not be able to handle it. There will be frame losses even if the transmission is error-free. The ***two common approaches for flow control are –***

- 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.

3.2 Error Detection and Correction:

Error Detection: Involves checking whether any error has occurred or not.

Error Correction: Involves exact number of bits corrupted and their locations.

The strategies used for Error Detection and Correction:

- Error-correcting codes (FEC (Forward Error Correction))
- Error-detecting codes.

Error-correcting codes:

1. Hamming codes.
2. Binary convolution codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

1. Hamming codes:

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

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

$$2^r \geq m + r + 1$$

where, r = redundant bit, m = data bit

Ex: Suppose the number of data bits is 7, then the number of redundant bits can be calculated using: = 2^4 ≥ 7 + 4 + 1 Thus, the number of redundant bits= 4.

Calculation of Hamming Distance

Hamming distance is the number of bit positions in which the two bits are different.

In order to calculate the Hamming distance between two strings, and , we perform their XOR operation, (a⊕ b), and then count the total number of 1s in the resultant string.

Example

Suppose there are two strings 1101 1001 and 1001 1101.

11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.

Example: Encode the data or message bits 0011 into 7 bit even parity Hamming code.

Solution: Given message=0011

No.of.message bits=4

Calculate no.of.parity bits by formula **$2^p>=m+p+1$**

$2^3>=4+3+1$

8>=8

Hence no.of.parity bits=3.

Total no.of.bits=4+3=7

Decimal no.	2^2	2^1	2^0
1	0	0	**1**
2	0	**1**	0
3	0	**1**	**1**
4	**1**	0	0
5	**1**	0	**1**
6	**1**	**1**	0
7	**1**	**1**	**1**

Parity bits at position(1,3,5,7)

			2^3		2^1	2^0
7	6	5	4	3	2	1
m4	m3	m2	p3	m1	p2	p1
0	0	1	p3	1	p2	p1

P1=(1,3,5,7)=p1 1 1 0,hence p1=0

P2=(2,3,6,7)=p2 1 0 0,hence p2=1

P3=(4,5,6,7)=p3 1 0 0,hence p3=1

Error position:p3p2p1=110=6

Total message=0011110

After correcting=0111110

Example: determine which bit is in error in the even parity. Hamming coded character is 1000001 .

Solution: Given message=1000001

No.of.message bits=7

Calculate no.of.parity bits by formula **$2^p>=m+p+1$**

$2^4>=7+3+1$

16>=11

Hence no.of.parity bits=4.

Total no.of.bits=7+4=11

Thus 7 databits and 4 check bits.

In Hamming codes the bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on. The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits. The rest (3,5, 6, 7, 9, etc.) are filled up with the m data bits.

Bit location table:

			2^3				2^2		2^1	2^0
11	10	9	8	7	6	5	4	3	2	1
M11	M10	M9	P8	M7	M6	M5	P4	M3	P2	P1
1	0	0	1	0	0	0	0	1	0	0

Detecting error:

P1 parity bits checks for 1,3,5,7,9,11=p1 10001=hence p1=0

P2 parity bits checks for 2,3,6,7,10,11=p210001=hence p2=0

P4 parity bits checks for 4,5,6,7=p4000=hence p4=0

P8 parity bits checks for 8,9,10,11=p8001=hence p8=1

Error detected at p4p3p2p1=1000=8th position

Decimal no.	2^3	2^2	2^1	2^0
1	0	0	0	**1**
2	0	0	**1**	0
3	0	0	**1**	**1**
4	**0**	**1**	0	0
5	**0**	**1**	0	**1**
6	**0**	**1**	**1**	0
7	**0**	**1**	**1**	**1**
8	**1**	**0**	**0**	**0**
9	**1**	**0**	**0**	**1**
10	**1**	**0**	**1**	**0**
11	**1**	**0**	**1**	**1**

Total message=00100001001

After correcting=00110001001

2.Convolutional code:

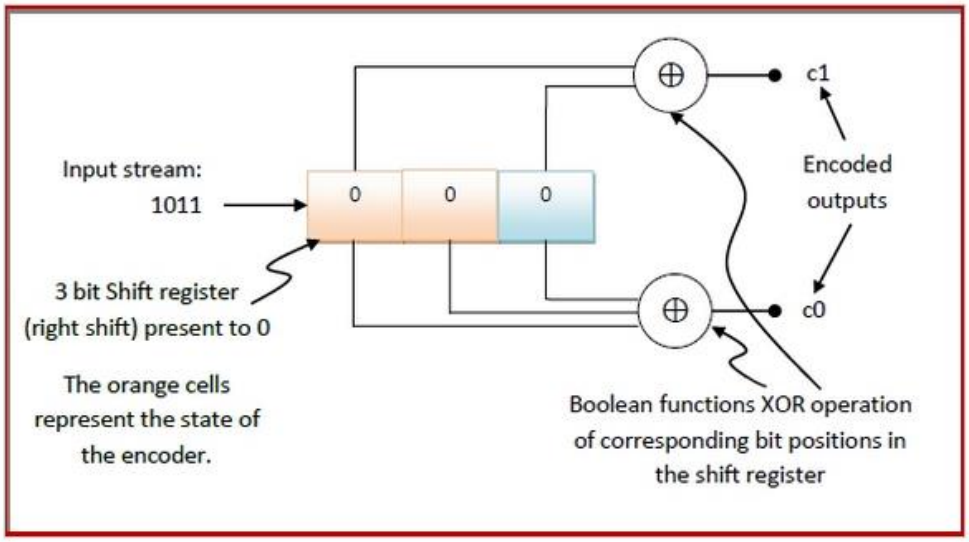
A convolutional code can be represented as (n,k, K) where

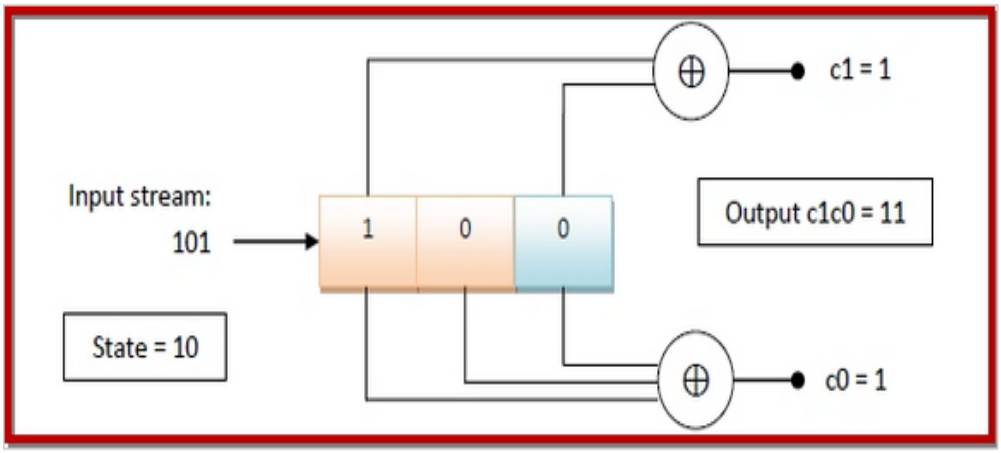
- *k* is the number of bits shifted into the encoder at one time. Generally, *k = 1*.
- n is the number of encoder output bits corresponding to k information bits.
- The code-rate, $R_c = k/n$.
- The encoder memory, a shift register of size k, is the constraint length.
- *n* is a function of the present input bits and the contents of *K*.
- The state of the encoder is given by the value of (*K - 1)* bits.

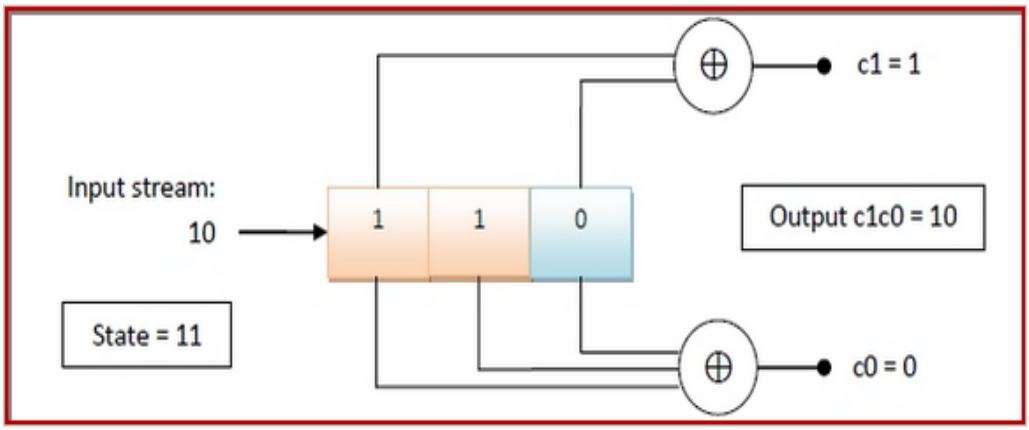
Example:

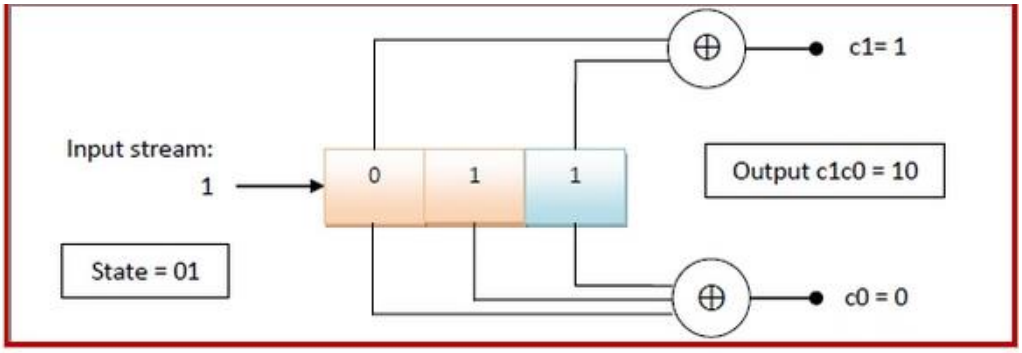
Let us consider a convolutional encoder with $k = 1$, $n = 2$ and K = 3.

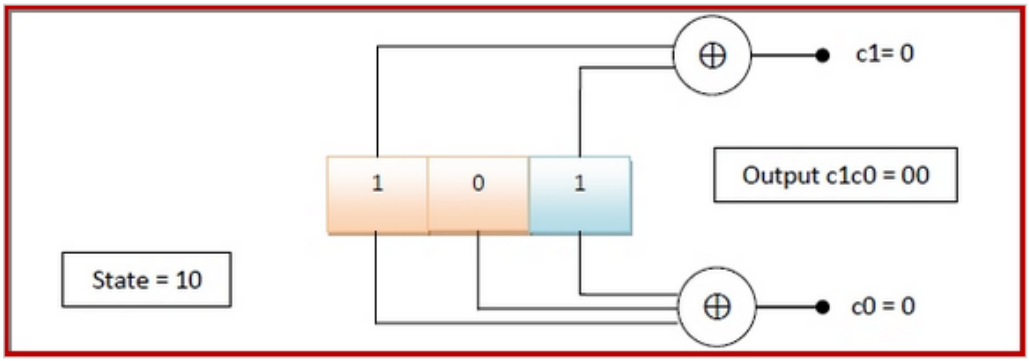
The code-rate, $R_c = k/n = 1/2$.


Input stream:
1011
0
0
0
c1
Encoded
outputs
c0
3 bit Shift register
(right shift) present to 0
The orange cells
represent the state of
the encoder.
Boolean functions XOR operation
of corresponding bit positions in
the shift register


Input stream:
101
1
0
0
c1 = 1
Output c1c0 = 11
State = 10
c0 = 1


Input stream:
10
1
1
0
c1 = 1
Output c1c0 = 10
State = 11
c0 = 0





For the binary convolution encoder given in the example –

The set of inputs = {0, 1}
The set of outputs = { 00, 10, 11}
The set of states = { 00, 01, 10, 11}

3. Reed-Solomon code:

- Reed-Solomon codes are linear block codes.
- Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on m bit symbols.
- Reed-Solomon codes are based on the fact that every n degree polynomial is uniquely determined by n + 1 points. For example, a line having the form ax + b is determined by two points.
- For m bit symbols, the codewords are 2m −1 symbols long.
- A Reed-Solomon code is specified as RS(*n,k*) with *s*-bit symbols.

  This means that the encoder takes *k* data symbols of *s* bits each and adds parity symbols to make an *n* symbol codeword. There are *n-k* parity symbols of s bits each. A Reed-Solomon decoder can correct up to *t* symbols that contain errors in a codeword, where *2t = n-k*.
- Example: A popular Reed-Solomon code is RS(255,223) with 8-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:
  n = 255, k = 223, s = 8
  2t = 32, t = 16
  The decoder can correct any 16 symbol errors in the code word: i.e. errors in up to 16 bytes anywhere in the codeword can be automatically corrected.

- Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. Reed-Solomon codes are used to correct errors in many systems including:

1. Storage devices (including tape, Compact Disk, DVD, barcodes, etc)
2. Wireless or mobile communications (including cellular telephones, microwave links, etc)
3. Satellite communications
4. Digital television / DVB
5. High-speed modems such as ADSL, xDSL, etc.

4. Low-Density Parity Check codes:

- Low - density parity check (LDPC) code is a linear error-correcting block code, suitable for error correction in large block sizes transmitted via very noisy channels.
- In an LDPC code, each output bit is formed from only a fraction of the input bits. This leads to a matrix representation of the code that has a low density of 1s,hence the name for the code.
- The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. This corrects errors.

3.3 Error Detecting Codes:

1. Parity.
2. Checksums.
3. Cyclic Redundancy Checks (CRCs).

1. Parity:

- A parity bit is an extra bit included in binary message to make total number of 1's either odd or even. Parity word denotes number of 1's in a binary string.
- Even Parity: Total number of 1's in the given data bit should be even. So if the total number of 1's in the data bit is odd then a single 1 will be appended to make total number of 1's even else 0 will be appended.
  For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100.
- odd parity system, if the total number of 1's in the given binary string (or data bits) are even then 1 is appended to make the total count of 1's as odd else 0 is appended.
  For example, With odd parity 1011010 becomes 10110101.

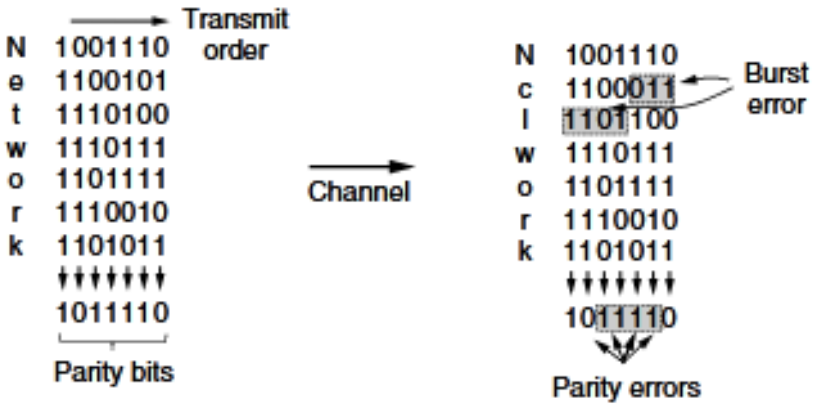
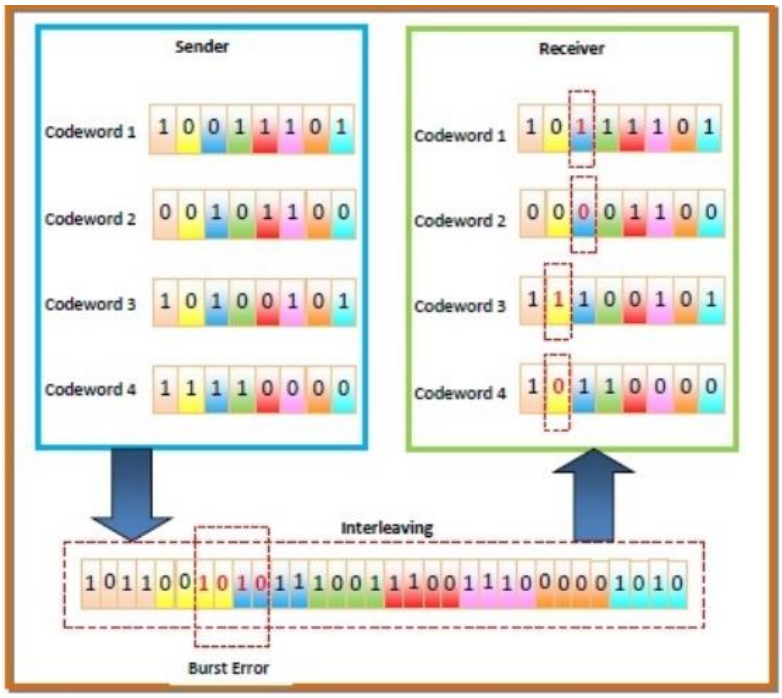
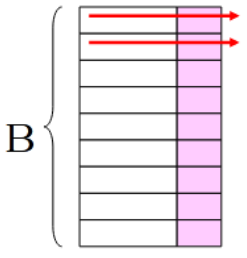


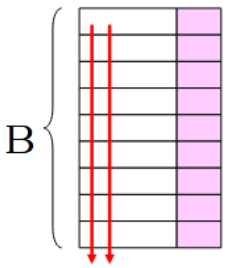
Figure 3-8. Interleaving of parity bits to detect a burst error.



Well, can we think of a way to turn a B-bit error burst into B single-bit errors?



Row-by-row transmission order



Col-by-col transmission order

Problem: Bits from a particular codeword are transmitted sequentially, so a B-bit burst produces multi-bit errors.

Solution: interleave bits from B different codewords. Now a B-bit burst produces 1-bit errors in B different codewords.

2. checksum:

- This is a block code method where it makes the use of **Checksum Generator** on Sender side and **Checksum Checker** on Receiver side.
- At the Sender side, the data is divided into equal subunits of n bit length by the checksum generator. This bit is generally of 16-bit length.
- These subunits are then added together using one's complement method. This sum is of n bits.
- The resultant bit is then complemented.
- This complemented sum which is called checksum is appended to the end of original data unit and is then transmitted to Receiver.
- The **Internet checksum**, also called the **IPv4 header checksum** is a checksum used in version 4 of the Internet Protocol (IPv4) to detect corruption in the header of IPv4 packets.

  - Convert the data segment into a series of 16-bit integers;
  - Calculate the sum of all 16-bit integers, allowing for the carry bit wrap-around;
  - Add the checksum to the final sum total;
  - If the final total is all 1's the data is validated;
  - If any 0's are detected the data has been corrupted.

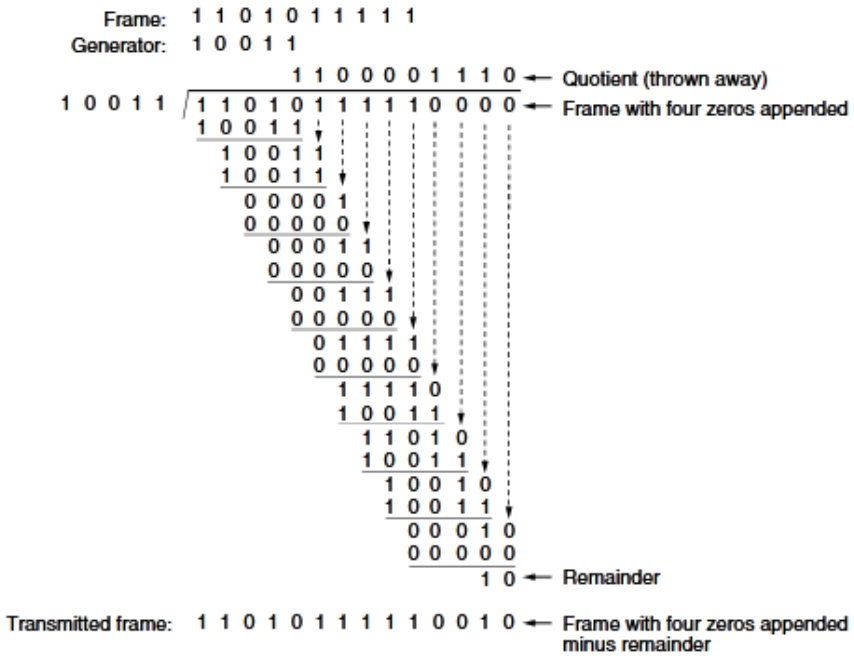
- **Fletcher's checksum**
  Fletcher checksum is an error – detection technique that uses two checksums to determine single-bit errors in a message transmitted over network channels.
  The **Fletcher checksum** is an algorithm for computing a position-dependent checksum.

3.CRC (Cyclic Redundancy Check):

- It is also known as a polynomial code.
- CRC uses **Generator Polynomial** which is available on both sender and receiver side.
- For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients 1, 1, 0, 0, 0, and 1: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x$

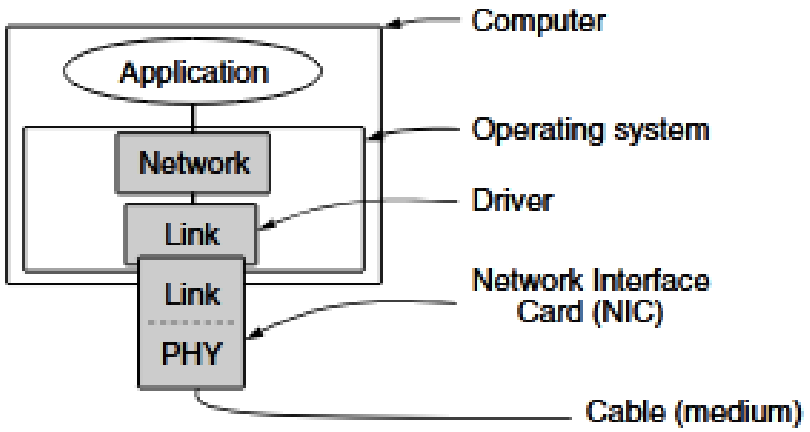
  The algorithm for computing the CRC is as follows:

  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 $x^r M(x)$.
  2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^r M(x)$, using modulo 2 division.
  3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^r M(x)$using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial T(x).



3.3 Elementary Data Link Protocols

Implementation of the physical, data link, and network layers:



- We assume that the physical layer, data link layer, and network layer are independent processes that communicate by passing messages back and forth.
- The physical layer process and some of the data link layer process run on dedicate hardware called a NIC (Network Interface Card).
- The rest of the link layer process and the network layer process run on the main CPU as part of the operating system, with the software for the link layer process often taking the form of a device driver.
- Other Possible Implementations:
  - Three processes offloaded to dedicated hardware called a network accelerator.
  - Three processes running on the main CPU on a software-defined ratio.

```
#define MAX PKT 1024                                    /* determines packet size in bytes * /
typedef enum {false, true} boolean;                     /* boolean type * /
typedef unsigned int seq nr;                            /* sequence or ack numbers * /
typedef struct {unsigned char data[MAX PKT];} packet;   /* packet definition * /
typedef enum {data, ack, nak} frame kind;               /* frame kind definition * /
typedef struct {                                        /* frames are transported in this layer * /
frame kind kind;                                        /* what kind of frame is it? * /
seq nr seq;                                             /* sequence number * /
seq nr ack;                                             /* acknowledgement number * /
packet info;                                            /* the network layer packet * /
} frame;
/* Wait for an event to happen; return its type in event. * /
void wait for event(event type * event);
/* Fetch a packet from the network layer for transmission on the channel. * /
void from network layer(packet * p);
/* Deliver information from an inbound frame to the network layer. * /
void to network layer(packet * p);
/* Go get an inbound frame from the physical layer and copy it to r. * /
void from physical layer(frame * r);
/* Pass the frame to the physical layer for transmission. * /
void to physical layer(frame * s);
/* Start the clock running and enable the timeout event. * /
void start timer(seq nr k);
/* Stop the clock and disable the timeout event. * /
void stop timer(seq nr k);
/* Start an auxiliary timer and enable the ack timeout event. * /
void start ack timer(void);
/* Stop the auxiliary timer and disable the ack timeout event. * /
void stop ack timer(void);
/* Allow the network layer to cause a network layer ready event. * /
void enable network layer(void);
/* Forbid the network layer from causing a network layer ready event. * /
void disable network layer(void);
/* Macro inc is expanded in-line: increment k circularly. * /
#define inc(k) if (k < MAX SEQ) k = k + 1; else k = 0
```

Figure:Some definitions needed in the protocols to follow. These definitions are located in the file protocol.h.

3.3.1 A Utopian Simplex Protocol:

- Data are transmitted in one direction only.
- Both transmitting and receiving network layers are always ready.

- Processing time can be ignored. Infinite buffer space is available.
- Since this protocol is totally unrealistic, it is often called Utopian Simplex protocol.

Implementation:

```
typedef enum {frame arrival} event type;
#include "protocol.h"
void sender1(void)
{
frame s;                                  /* buffer for an outbound frame * /
packet buffer;                            /* buffer for an outbound packet * /
while (true) {
from network layer(&buffer);              /* go get something to send * /
s.info = buffer;                          /* copy it into s for transmission * /
to physical layer(&s);                    /* send it on its way * /
}
 /* Tomorrow, and tomorrow, and tomorrow,
Creeps in this petty pace from day to day
To the last syllable of recorded time.
– Macbeth, V, v * /
}
void receiver1(void)
{
frame r;
event type event;                         /* filled in by wait, but not used here * /
while (true) {
wait for event(&event);                   /* only possibility is frame arrival * /
from physical layer(&r);                  /* go get the inbound frame * /
to network layer(&r.info);                /* pass the data to the network layer * /
}
}
```

- This protocol has two different procedures, a sender and a receiver.
- MAX_SEQ is not needed because no sequence numbers or acknowledgements are used.
- The utopia protocol is unrealistic because it does not handle either flow control or error correction.

3.3.2 A Simplex Stop and Wait Protocol for an Error Free Channel:

Problem: Preventing the sender from flooding the receiver with frames faster than the latter is able to process them. The communication channel is still assumed to be error free, however, and the data traffic is still simplex.

Solution: To build the receiver to be powerful enough to process a continuous stream of back-to-back frames (or, equivalently, define the link layer to be slow enough that the receiver can keep up). It must have sufficient buffering and processing abilities to run at

the line rate and must be able to pass the frames that are received to the network layer quickly enough.

Drawbacks: It requires dedicated hardware and can be wasteful of resources if the utilization of the link is mostly low.

Another Solution: is to have the receiver provide feedback to the sender. After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to transmit the next frame. After having sent a frame, the sender is required by the protocol to bide its time until the little dummy (i.e., acknowledgement) frame arrives. This delay is a simple example of a flow control protocol.

Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called stop-and-wait.

Implementation:

```
typedef enum {frame arrival} event type;
#include "protocol.h"
void sender2(void)
{
frame s;                          /* buffer for an outbound frame * /
packet buffer;                    /* buffer for an outbound packet * /
event type event;                 /* frame arrival is the only possibility * /
while (true) {
from network layer(&buffer);      /* go get something to send * /
s.info = buffer;                  /* copy it into s for transmission * /
to physical layer(&s);            /* bye-bye little frame * /
wait for event(&event);           /* do not proceed until given the go ahead * /
}
}
void receiver2(void)
{
frame r, s;                       /* buffers for frames * /
event type event;                 /* frame arrival is the only possibility * /
while (true) {
wait for event(&event);           /* only possibility is frame arrival * /
from physical layer(&r);          /* go get the inbound frame * /
to network layer(&r.info);        /* pass the data to the network layer * /
to physical layer(&s);            /* send a dummy frame to awaken sender * /
}
}
```

3.3.3 A Simplex Stop and Wait Protocol for a Noisy Channel:

- Noisy channel protocols are commonly known as communication protocols.

- This protocol takes into account the facts that the receiver has a finite processing speed and a finite buffer capacity, so the protocol must explicitly prevent the sender from flooding the receiver with data faster than it can be handled.

Protocol 2 working: ***A Simplex Stop and Wait Protocol for an Error Free Channel***

Adding a timer. The sender could send a frame, but the receiver would only send an acknowledgement frame if the data were correctly received. If a damaged frame arrived at the receiver, it would be discarded. After a while the sender would timeout and sends the frame again. This process would be repeated until the frame finally arrived intact.

Problem:

Consider the following scenario:

1. The network layer on A gives packet 1 to its data link layer. The packet is correctly received at B and passed to the network layer on B. B sends an acknowledgement frame back to A.
2. The acknowledgement frame gets lost completely. It just never arrives at all. Life would be a great deal simpler if the channel mangled and lost only data frames and not control frames, but sad to say,the channel is not very discriminating.
3. The data link layer on A eventually times out. Not having received an acknowledgement, it (incorrectly) assumes that its data frame was lost or damaged and sends the frame containing packet 1 again.
4. The duplicate frame also arrives intact at the data link layer on B and is unwittingly passed to the network layer there. If A is sending a file to B, part of the file will be duplicated (i.e., the copy of the file made by B will be incorrect and the error will not have been detected). In other words, the protocol will fail.

    - ***Duplication of Frames takes place.***

Solution: Assign the frames with sequence numbers.

- It's simply known as Automatic Repeat Request (Stop – and –Wait ARQ) protocol.

Implementation:

```
#define MAX SEQ 1                          /* must be 1 for protocol 3 * /
typedef enum {frame arrival, cksum err, timeout} event type;
#include "protocol.h"
void sender3(void)
{
seq nr next frame to send;          /* seq number of next outgoing frame * /
frame s;                            /* scratch variable * /
packet buffer;                      /* buffer for an outbound packet * /
event type event;
next frame to send = 0;             /* initialize outbound sequence numbers * /
from network layer(&buffer);        /* fetch first packet * /
while (true) {
s.info = buffer;                    /* construct a frame for transmission * /
s.seq = next frame to send;         /* insert sequence number in frame * /
```

```
to physical layer(&s);                  /* send it on its way * /
start timer(s.seq);                     /* if answer takes too long, time out * /
wait for event(&event);                 /* frame arrival, cksum err, timeout * /
if (event == frame arrival) {
from physical layer(&s);                /* get the acknowledgement * /
if (s.ack == next frame to send) {
stop timer(s.ack);                      /* turn the timer off * /
from network layer(&buffer);                  /* get the next one to send * /
inc(next frame to send);                /* invert next frame to send * /
}
}
}
}
void receiver3(void)
{
seq nr frame expected;
frame r, s;
event type event;
frame expected = 0;
while (true) {
wait for event(&event);                 /* possibilities: frame arrival, cksum err * /
if (event == frame arrival) {                 /* a valid frame has arrived * /
from physical layer(&r);                /* go get the newly arrived frame * /
if (r.seq == frame expected) {          /* this is what we have been waiting for * /
to network layer(&r.info);              /* pass the data to the network layer * /
inc(frame expected);                    /* next time expect the other sequence nr * /
}
s.ack = 1 − frame expected;             /* tell which frame is being acked * /
to physical layer(&s);                  /* send acknowledgement * /
}
}
}
```

(Sender3 Function:

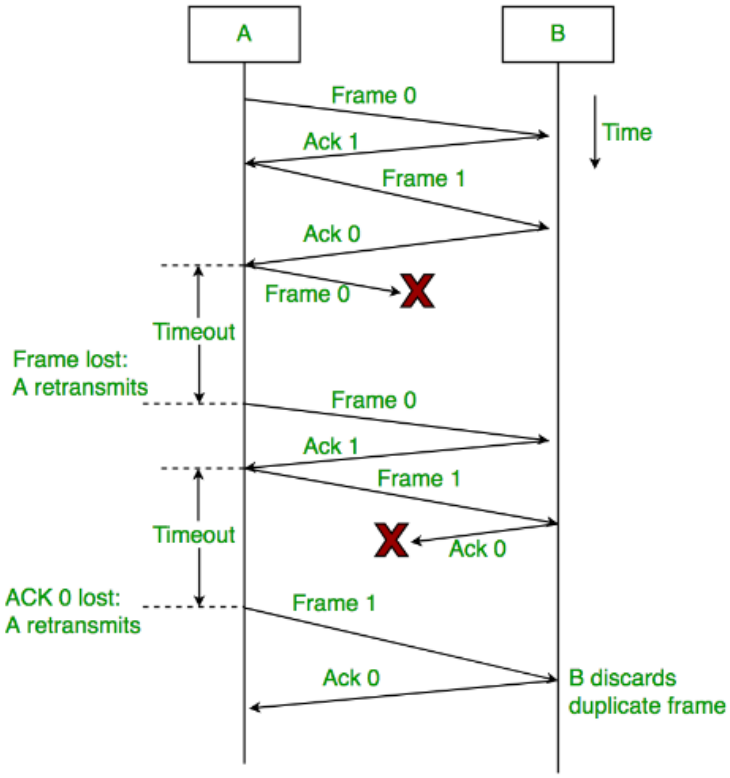
1. Initialize variables.
2. Retrieve a packet from the network layer.
3. Construct a frame for transmission.
4. Send the frame to the physical layer.
5. Start a timer for the frame.
6. Wait for an event (frame arrival, checksum error, or timeout).
7. If the event is a frame arrival: a. Receive an acknowledgment from the physical layer. b. Check if the acknowledgment matches the expected frame. c. If it matches, stop the timer and get the next packet from the network layer. d. Update the next frame to send.

8. Repeat the loop.

Receiver3 Function:

1. Initialize variables.
2. Wait for an event (frame arrival or checksum error).
3. If the event is a frame arrival: a. Receive a frame from the physical layer. b. Check if the received frame's sequence number matches the expected frame. c. If it matches, pass the data to the network layer. d. Update the expected frame. e. Construct an acknowledgment frame. f. Send the acknowledgment to the physical layer.
4. Repeat the loop.

)



3.4 Sliding Window Protocols:

- In the previous protocols, data frames were transmitted in one direction only but this protocol supports data transmission in both the directions.
- Simplex Mode:
  Each using a separate link for simplex data traffic (in different directions). Each link is then comprised of a ‘‘forward’’ channel (for data) and a ‘‘reverse’’ channel (for acknowledgements). In both cases the capacity of the reverse channel is almost entirely wasted.
- Full-duplex Mode:
  Reverse channel normally has the same capacity as the forward channel. In this model the data frames from A to B are intermixed with the acknowledgement

frames from A to B. By looking at the kind field in the header of an incoming frame, the receiver can tell whether the frame is data or an acknowledgement.

- Piggybacking:

  When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet. The acknowledgement is attached to the outgoing data frame (using the ack field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking.

  Advantages:

  - The principal advantage of using piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth.
  - The ack field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum.
  - In addition, fewer frames sent generally means a lighter processing load at the receiver.

  Disadvantage:

  How long should the data link layer wait for a packet onto which to piggyback the acknowledgement? If the data link layer waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements.

  Working Principle:

  - In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window.
  - The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to 2^n-1.
  - The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.
  - The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

    Example:

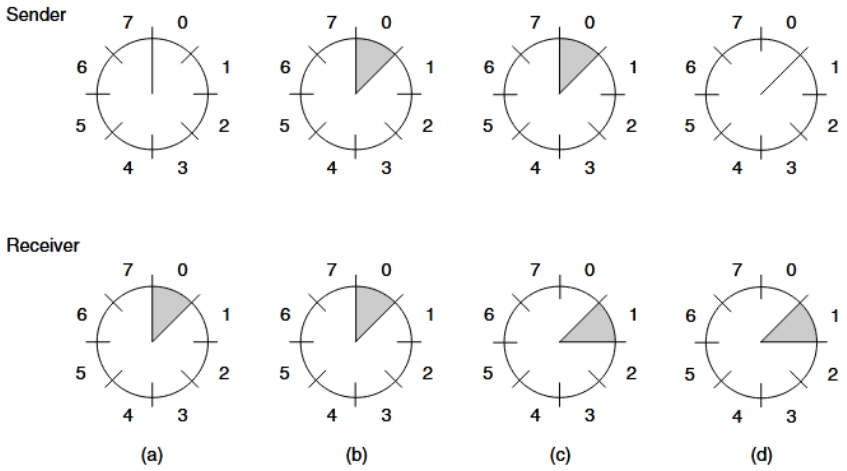


Figure 3-15. A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

1. A sender has not yet set sender window but receiver is accepting frame 0.
2. Sender has sent frame 0 but not acknowledged, Fame 0 is not been received by receiver yet, hence still waiting for frame 0.
3. Sender has opened the window for frame 0(not acknowledged) and receiver has received frame 0,hence it advances receiver window to accept frame 1.
4. Sender now received acknowledgment to frame 0,hence closes its window and receiver is still waiting for frame1.

Different Sliding Window Protocols:

1. A One-Bit Sliding Window Protocols:

- Window size=1
- Stop and wait Protocol
- Sender transmits a frame and waits for ack before transmitting a next frame.

```
define MAX SEQ 1                    /* must be 1 for protocol 4 * /
typedef enum {frame arrival, cksum err, timeout} event type;
#include "protocol.h"
void protocol4 (void)
{
seq nr next frame to send;          /* 0 or 1 only * /
seq nr frame expected;              /* 0 or 1 only * /
frame r, s;                         /* scratch variables * /
packet buffer;                      /* current packet being sent * /
event type event;
next frame to send = 0;             /* next frame on the outbound stream * /
```

```
frame expected = 0;                    /* frame expected next * /
from network layer(&buffer);      /* fetch a packet from the network layer * /
s.info = buffer;                       /* prepare to send the initial frame * /
s.seq = next frame to send;          /* insert sequence number into frame * /
s.ack = 1 − frame expected;         /* piggybacked ack * /
to physical layer(&s);             /* transmit the frame * /
start timer(s.seq);                /* start the timer running * /
while (true) {
wait for event(&event);            /* frame arrival, cksum err, or timeout * /
if (event == frame arrival) {        /* a frame has arrived undamaged * /
from physical layer(&r);             /* go get it * /
if (r.seq == frame expected) {                /* handle inbound frame stream * /
to network layer(&r.info);                    /* pass packet to network layer * /
inc(frame expected);                          /* invert seq number expected next * /
}
if (r.ack == next frame to send) {              /* handle outbound frame stream * /
stop timer(r.ack);                                /* turn the timer off * /
from network layer(&buffer);                    / * fetch new pkt from network layer * /
inc(next frame to send);                      /* invert sender's sequence number * /
}
}
s.info = buffer;                          /* construct outbound frame * /
s.seq = next frame to send;                 /* insert sequence number into it * /
s.ack = 1 − frame expected;                 /* seq number of last received frame * /
to physical layer(&s);                        /* transmit a frame * /
start timer(s.seq);                           /* start the timer running * /
}
}
```

Figure : A 1-bit sliding window protocol

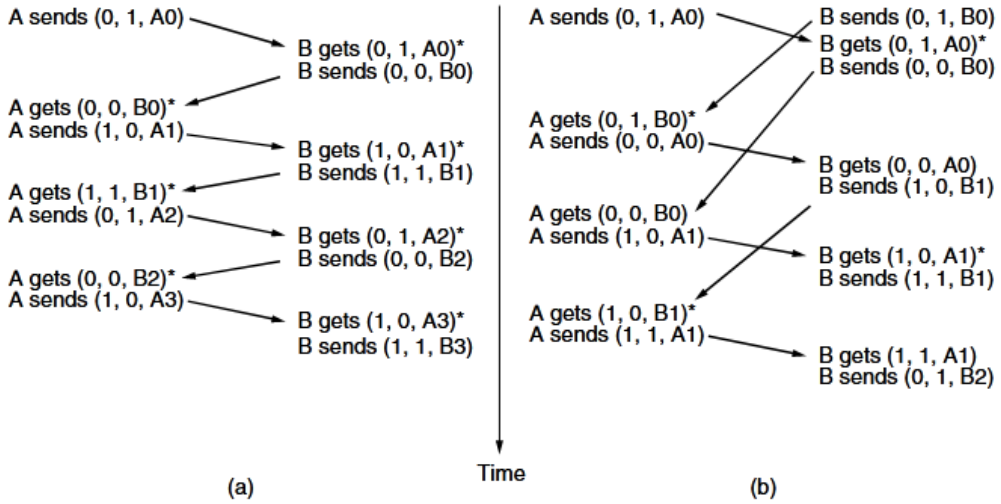


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

Normal Case:

- If no simultaneous transmission, then its normal case.
- Assume sender A sends frame with seq no 0, previously accepted ack 1 and packet no A0 within that frame.
- * indicates the packet as accepted. And B received and accepted the A0.
- Now B sends frame with seq no 0, previously accepted ack 0 and packet no B0 within that frame.
- After that A sends a next frame with seq no 1, previously accepted ack 0 and packet no A1 within that frame.
- No duplicate frames and no wastage of bandwidths.

Abnormal Case:

- If simultaneous transmission, then its abnormal case.
- (timers are of different length)Assume sender A sends frame with seq no 0, previously accepted ack 1 and packet no A0 within that frame.At the same time B also sends frame with seq no 0, previously accepted ack 1 and packet no B0 within that frame.
- Sender B receives A0 and sends frame with seq no 0(resends same frame since ack is not received for B0), ack 1 and packet no B0 within that frame.
- Sender A accepts B0 since it is new frame. It sends (0,0,A0)-0 retransmission since ack for A0 is not received,0-ACK for B0,A0-same packet.
- B already received frame with seq no 0,hence discarded.Since B has received ack,it sends new frame as (1,0,B1).
- Retransmission of packets takes place.

2.

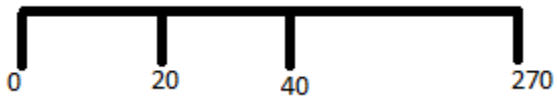
3. **A Protocol Using Go-Back N:**

It is assumed that transmission time for sender and receiver is negligible but in many situations it is not true. There is a Round Trip Time that effects on bandwidth utilization and efficiency of communication channel.

Example:

- 50 kbps satellite channel
- 500 msec round-trip propagation delay.
- Lets use Protocol 4 (previous) to send a 1000 bit frames via the satellite.
- t = 0: the sender starts the first frame
- t = 20 msec: the frame has been completely sent.
- t = 270 msec: the frame has fully arrived at the satellite.
- t = 520 msec: the acknowledgment has arrived at the sender.
- Sender was blocked 500/520 or 96% of the time or to send the packet the sender utilized 4% of the available bandwidth.

For 50kbps, to transmit 1000 bits it takes time of 1000/50=20msec.
Since 500msec is propagation delay, it takes 250msec for one way.



270(250+20)
Ack(270+250=520msec)
96% sender does not do anything,only 4% was utilized.

- The problem?
  - Consequence of the rule that requires a sender to wait for an acknowledgment before sending another frame.
  - Relaxing this condition will enable achieving significantly better throughput.
- Solution!
  - Allowing sender to transmit *w* frames before blocking.
  - Acknowledgment will arrive for previous frames before the window becomes full.

- Must correctly establish what is the actual size of w.
  - Number of frames to fit inside the channel.
- Capacity of the channel is determined by
  - Bandwidth in bits/sec => B, multiplied by
  - One-way transit time => D, or
  - The bandwidth-delay product of the link: BD
- The actual size should be set to : $w = 2BD + 1$

- Example:
  - Link with the bandwidth of 50 kbps
  - One way transit time of 250 msec
  - Bandwidth delay product is
  - BD = 50 kbps x 250 msec = 12.5 kbits or 12.5 frames of 1000 bits.
  - 2BD+1 = 26 frames.
  - t = 0: sends a first frame and subsequent frames after 20 msec.
  - t = 520 msec: acknowledgment for the first frame is received, while 26 frames were transmitted.
  - Thereafter, acknowledgments will arrive every 20 msec.

For 1 frame=20msec

For 26 frames=20*26=520msec.

  - Just in time for the sender to continue transmitting.
  - 25 or 26 unacknowledged frames will always be outstanding.
  - Hence, the senders maximum window size is 26 frames.
- For smaller window size, the utilization of the link will be less than 100%.
- Upper bound of the link utilization:
  - LU< = w/(1+2BD)
  - It does not allow for any frame processing time
  - Treats acknowledgment frame as having zero length.

- From the equation:
  - Large bandwidth-delay product requires a large window.
  - With stop-and-wait for which w = 1, if there is even one frame's worth of propagation delay, the efficiency will be less than 50% as sender will have to wait for D time (= 1 frame gen) before sending the next frame until the ack arrives
- **Pipelining** – technique where the number of frames (much greater than 1) are sent immediately.
- What will happen when a certain frame is received in error?!

Solution: a.Go-Back N Protocol, b. Selective Repeat

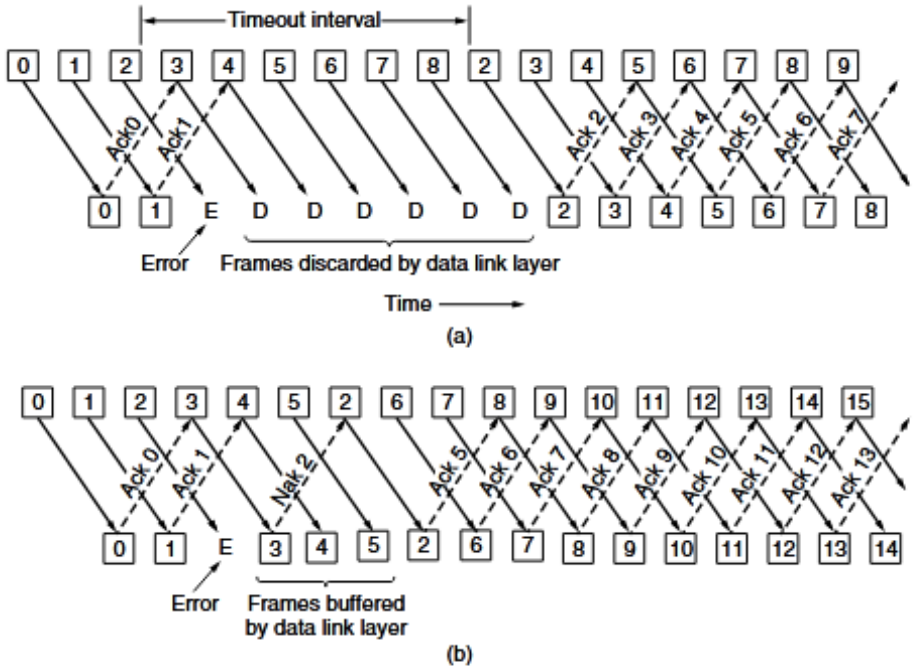


Figure 3-18. Pipelining and error recovery. Effect of an error when (a) receiver's window size is 1 and (b) receiver's window size is large.

Selective Repeat Protocol:

Cumulative Acknowledgment:

- When an acknowledgment comes in for frame n, frames n-1, n-2, ... are also **automatically acknowledged**.
- Because the Go-back-n protocol has multiple outstanding frames, it logically needs **multiple timers**, one per outstanding frame.
  - Each frame times out **independently** of all the other ones.
  - The pending timeouts form a **linked list**, with each node of the list containing the
    - Number of clock ticks until the timer expires,
    - Frame being timed,
    - A pointer to the next node

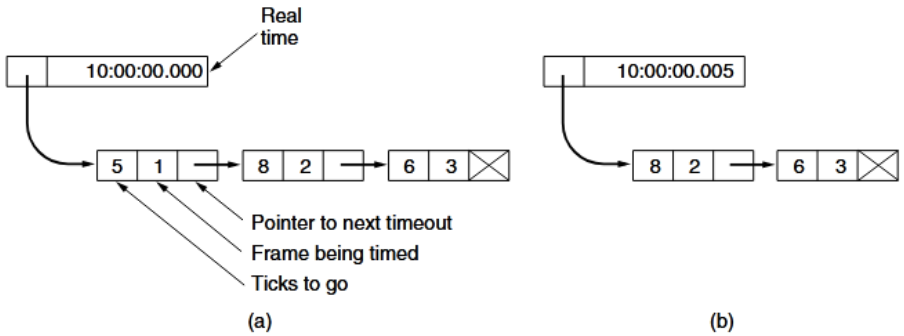


Figure 3-20. Simulation of multiple timers in software. (a) The queued timeouts. (b) The situation after the first timeout has expired.

As an illustration of how the timers could be implemented, consider the example of Fig. 3-20(a). Assume that the clock ticks once every 1 msec. Initially,

- The go-back-n protocol works well if errors are rare.
- If the line is poor it wastes a lot of bandwidth on retransmitting frames.
- A **Selective Repeat** protocol allows the receiver to accept and buffer the frames following a damaged or lost one.

- Sender and Receiver maintain a window of outstanding and acceptable sequence numbers, respectively.
- Sender:
  - Window size starts at 0,
  - Grows to some predefined maximum,
- Receiver:
  - Is always fixed in size
  - Equal to the predetermined maximum.
  - Has a buffer reserved for each sequence number within its fixed window.
  - Associated with it is a bit (arrived) telling whether the buffer is full or empty.

- Whenever the frame arrives, the receiver does the following:
  - Check the frame sequence number if it does fall within its fixed window,
  - If this is the case and the sequence has not been received in previous transmissions, it will be stored.
  - It does so without regard if the frame contains the next packet expected by the network layer.
  - This frame is going to be kept until all the lower-numbered frames have been delivered to the network layer in the correct order.

- Nonsequential receive introduces further constraints on frame sequence numbers compared to the protocols that accepted frames in order.
  - Example:
    - 3-bit sequence number,
    - 7 frames are permitted to be sent by transmitter, before it is required to wait (for acknowledgment).
  - Window size restriction:
    - Must be at most half the range of the sequence numbers.
    - Window size for the Selective Repeat protocol is (MAX_SEQ+1)/2.

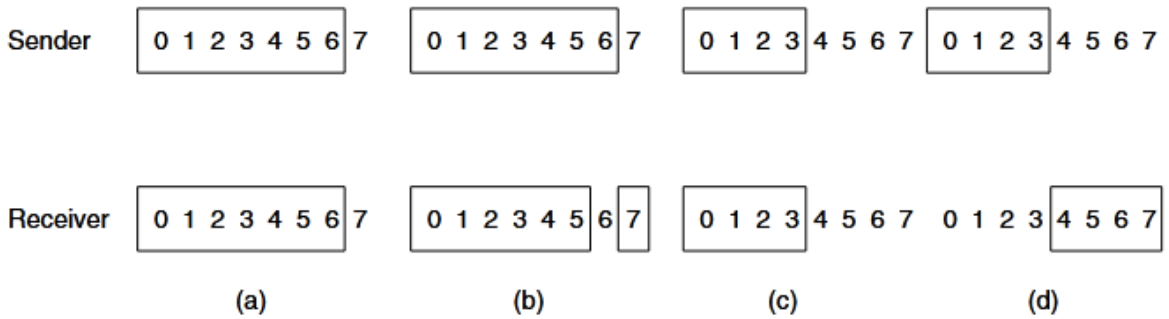


Figure 3-22. (a) Initial situation with a window of size7. (b) After 7 frames have been sent and received but not acknowledged. (c) Initial situation with a window size of 4. (d) After 4 frames have been sent and received but not acknowledged.

Implementation:

```
#define MAX SEQ 7                    /* should be 2ˆn − 1 * /
#define NR BUFS ((MAX SEQ + 1)/2)
```

```
typedef enum {frame arrival, cksum err, timeout, network layer ready, ack timeout}
        event type;
#include "protocol.h"
boolean no nak = true;                          /* no nak has been sent yet * /
seq nr oldest frame = MAX SEQ + 1;              /* initial value is only for the simulator * /
static boolean between(seq nr a, seq nr b, seq nr c)
{
/* Same as between in protocol 5, but shorter and more obscure. * /
return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}
static void send frame(frame kind fk, seq nr frame nr, seq nr frame expected, packet
        buffer[ ])
{
/* Construct and send a data, ack, or nak frame. * /
frame s;                                /* scratch variable * /
s.kind = fk;                            /* kind == data, ack, or nak * /
if (fk == data) s.info = buffer[frame nr % NR BUFS];
s.seq = frame nr;                       /* only meaningful for data frames * /
s.ack = (frame expected + MAX SEQ) % (MAX SEQ + 1);
if (fk == nak) no nak = false;          /* one nak per frame, please * /
to physical layer(&s);                  /* transmit the frame * /
if (fk == data) start timer(frame nr % NR BUFS);
stop ack timer();                       /* no need for separate ack frame * /
}
void protocol6(void)
{
seq nr ack expected;                    /* lower edge of sender's window * /
seq nr next frame to send;              /* upper edge of sender's window + 1 * /
seq nr frame expected;                  /* lower edge of receiver's window * /
seq nr too far;                         /* upper edge of receiver's window + 1 * /
int i;                                  /* index into buffer pool * /
frame r;                                /* scratch variable * /
packet out buf[NR BUFS];                /* buffers for the outbound stream * /
packet in buf[NR BUFS];                 /* buffers for the inbound stream * /
boolean arrived[NR BUFS];               /* inbound bit map * /
seq nr nbuffered;                       / * how many output buffers currently used * /
event type event;
enable network layer();                 /* initialize * /
ack expected = 0;                       /* next ack expected on the inbound stream * /
next frame to send = 0;                 /* number of next outgoing frame * /
frame expected = 0;
too far = NR BUFS;
```

```
nbuffered = 0;                          / * initially no packets are buffered * /
for (i = 0; i < NR BUFS; i++) arrived[i] = false;
while (true) {
wait for event(&event);                 /* five possibilities: see event type above * /
switch(event) {
case network layer ready:               /* accept, save, and transmit a new frame * /
nbuffered = nbuffered + 1;              / * expand the window * /
from network layer(&out buf[next frame to send % NR BUFS]);    /* fetch new packet * /
send frame(data, next frame to send, frame expected, out buf); / * transmit the frame * /
inc(next frame to send);                /* advance upper window edge * /
break;
case frame arrival:                     /* a data or control frame has arrived * /
from physical layer(&r);                /* fetch incoming frame from physical layer * /
if (r.kind == data) {
/* An undamaged frame has arrived. * /
if ((r.seq != frame expected) && no nak)
send frame(nak, 0, frame expected, out buf); else start ack timer();
if (between(frame expected,r.seq,too far) && (arrived[r.seq%NR BUFS]==false)) {
/* Frames may be accepted in any order. * /
arrived[r.seq % NR BUFS] = true;        /* mark buffer as full * /
in buf[r.seq % NR BUFS] = r.info;       /* insert data into buffer * /
while (arrived[frame expected % NR BUFS]) {
/* Pass frames and advance window. * /
to network layer(&in buf[frame expected % NR BUFS]);
no nak = true;
arrived[frame expected % NR BUFS] = false;
inc(frame expected);                    /* advance lower edge of receiver's window * /
inc(too far);                           /* advance upper edge of receiver's window * /
start ack timer();                      /* to see if a separate ack is needed * /
}
}
}
if((r.kind==nak) && between(ack expected,(r.ack+1)%(MAX SEQ+1),next frame to send))
send frame(data, (r.ack+1) % (MAX SEQ + 1), frame expected, out buf);
while (between(ack expected, r.ack, next frame to send)) {
nbuffered = nbuffered − 1;               /* handle piggybacked ack * /
stop timer(ack expected % NR BUFS);      /* frame arrived intact * /
inc(ack expected);                      /* advance lower edge of sender's window * /
}
break;
case cksum err:
if (no nak) send frame(nak, 0, frame expected, out buf);    /* damaged frame * /
```

```
break;
case timeout:
send frame(data, oldest frame, frame expected, out buf);          / * we timed out * /
break;
case ack timeout:
send frame(ack,0,frame expected, out buf);          / * ack timer expired; send ack * /
}
if (nbuffered < NR BUFS) enable network layer(); else disable network layer();
}
}
```

Figure : A sliding window protocol using selective repeat.

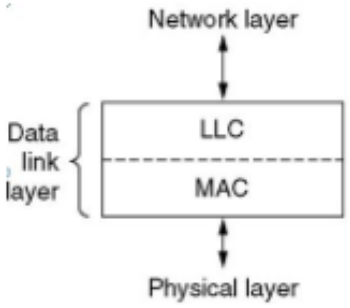
https://www.youtube.com/playlist?list=PLNkOUFO46yCKuv4763gEPF9TIORXG-qI_

The medium access control sublayer:

- Network links can be divided into two categories: those using point-to-point
- connections and those using broadcast channels.
- In any broadcast network, the key issue is how to determine who gets to use
- the channel when there is competition for it.
- Broadcast channels are sometimes referred to as multi-access channels or random access channels.
- To coordinate the access to the channel, multiple access protocols are requiring.
- All these protocols belong to the MAC sub layer. Data Link layer is divided into two

sub layers:

1. Logical Link Control (LLC)- is responsible for error control & flow control.
2. Medium Access Control (MAC)- MAC is responsible for multiple access resolutions.



THE CHANNEL ALLOCATION PROBLEM:

In broadcast networks, single channel is shared by several stations. This channel can be allocated to only one transmitting user at a time. There are two different methods of channel allocations:

1. Static Channel Allocation- a single channel is divided among various users either on the basis of frequency (FDM) or on the basis of time (TDM). In FDM, fixed frequency is assigned to each user, whereas, in TDM, fixed time slot is assigned to each user.

- If there are N users, the bandwidth is divided into N equal sized partitions, where each user is assigned with one portion. This is because, each user has a private frequency band.
- When there is only a small and constant number of users, each user has a heavy load of traffic, this division is a simple and efficient allocation mechanism.
- Let us take a wireless example of FM radio stations, each station gets a portion of FM band and uses it most of the time to broadcast its signal.
- When the number of senders is large and varying or traffic is suddenly changing, FDM faces some problems.
- If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. And if more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.
- A static allocation is a poor fit to most computer systems, in which data traffic is extremely burst, often with peak traffic to mean traffic ration of 1000:1, consequently most of the channels will be idle most of the time.
- The poor performance of static FDM can easily be seen with simple queueing theory calculation.

✓ Let us start with mean time delay T,
✓ Send a frame onto a channel of capacity C bps.
✓ Let us assume frames arrive randomly at an average arrival time λ frames/sec
✓ Average length of the frame is 1/μ bits.
✓ With the help of these parameters the service rate of channel is μC frame/sec
✓ Standard queueing theory result is −
T= 1/(μC- λ)
✓ Now divide the single channel into N independent subchannels, each with capacity C/N bps.
✓ The mean input rate on each subchannel is λ/N.
✓ Finally we get
TN=1/(μ(C/N) – (λ/N))
= N/(μC- λ)
= NT

Example

If C is 100Mbps, the mean frame length 1/μ is 10,000 bits and the frame arrival time λ is 5000 frames/ sec then,
T=1/(μC- λ)
= 200 μsec.

2. Dynamic Channel Allocation-

- The technique in which channels are not permanently allocated to the users is called dynamic channel allocation. In this technique, no fixed frequency or fixed time slot is allotted to the user.
- The allocation depends upon the traffic. If the traffic increases, more channels are allocated, otherwise fewer channels are allocated to the users.
- This technique optimizes bandwidth usage and provides fast data transmission.

The following are the assumptions in dynamic channel allocation:

1. Independent Traffic. The model consists of N independent stations (e.g., computers, telephones), each with a program or user that generates frames for transmission. The expected number of frames generated in an interval of length Δt is $\lambda \Delta t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.
2. Single Channel. A single channel is available for all communication. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign them different roles (e.g., priorities).
3. Observable Collisions. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a collision. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No errors other than those generated by collisions occur.
4. Continuous or Slotted Time. Time may be assumed continuous, in which case frame transmission can begin at any instant. Alternatively, time may be slotted or divided into discrete intervals (called slots). Frame transmissions must then begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.
5. Carrier Sense or No Carrier Sense. With the carrier sense assumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

MULTIPLE ACCESS PROTOCOLS:

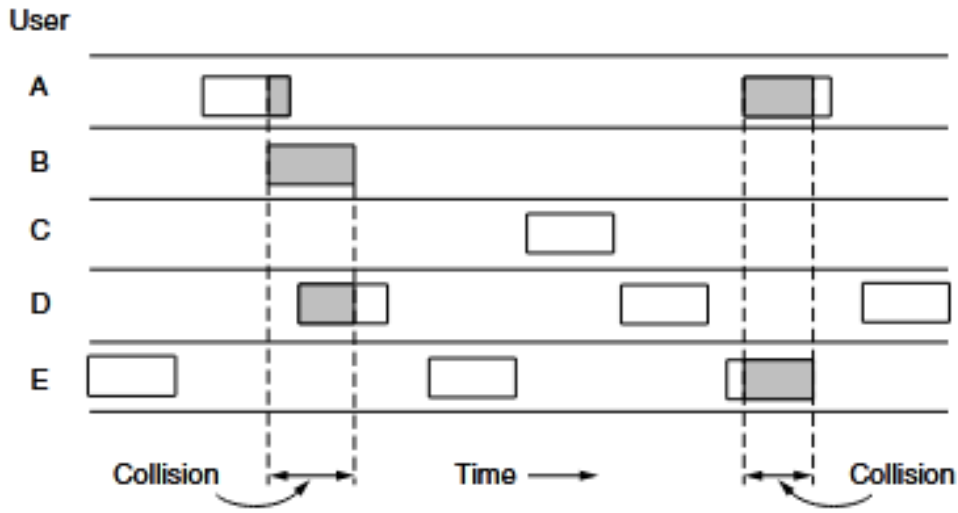
1. ALOHA:

ALOHA was developed at University of Hawaii in early 1970s by Norman Abramson. It was used for ground based radio broadcasting. In this method, stations share a common channel. When two stations transmit simultaneously, collision occurs and frames are lost. There are two different versions of ALOHA:

- Pure ALOHA
- Slotted ALOHA

Pure ALOHA:

In pure ALOHA, stations transmit frames whenever they have data to send. When two stations transmit simultaneously, there is collision and frames are lost. In pure ALOHA, whenever any station transmits a frame, it expects an acknowledgement from the receiver. If acknowledgement is not received within specified time, the station assumes that the frame has been lost. If the frame is lost, station waits for a random amount of time and sends it again. This waiting time must be random; otherwise, same frames will collide again and again. Whenever two frames try to occupy the channel at the same time, there will be collision and both the frames will be lost.



Analysis of Pure ALOHA:

Notation:

– Tf = frame time (processing, transmission, propagation)

– S: Average number of successful transmissions per Tf ; that is, the throughput or efficiency.

– G: Average number of total frames transmitted per Tf

– D: Average delay between the time a packet is ready for transmission and the completion of successful transmission.

The following assumptions are:

– All frames are of constant length

– The channel is noise-free; the errors are only due to collisions.

– Frames do not queue at individual stations

– The channel acts as a Poisson process.

Since S represents the number of “good” transmissions per frame time, and G

represents the total number of attempted transmissions per frame time, then we have

S = G ´ (Probability of good transmission)

• The vulnerable time for a successful transmission is 2Tf

• So, the probability of good transmission is not to have an "arrival" during the vulnerable time .

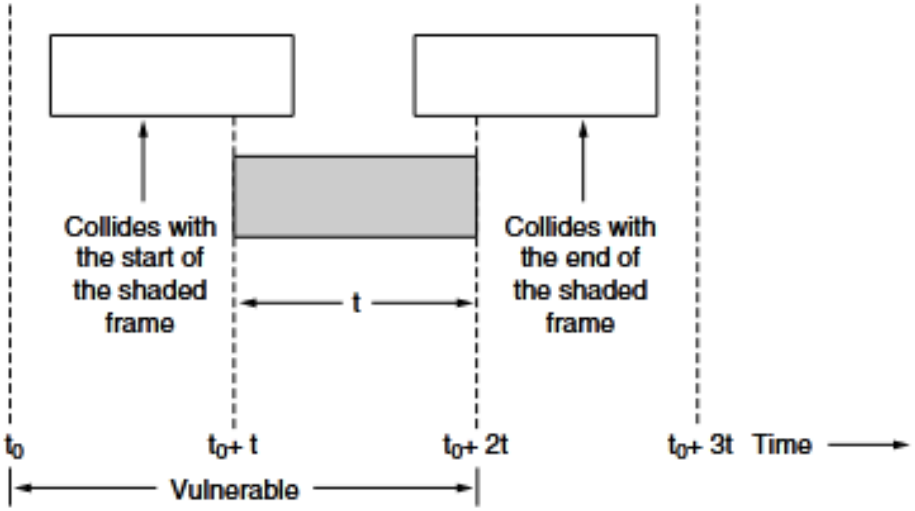


Figure 4-2. Vulnerable period for the shaded frame.

Using : $$P_k(t) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

And setting $t = 2T_f$ and $k = 0$, we get

$$P_0(2T_f) = \frac{(\lambda \cdot 2T_f)^0 e^{-\lambda 2T_f}}{0!} = e^{-2G}$$

becasue $\lambda = \frac{G}{T_f}$. Thus, $S = G \cdot e^{-2G}$

If we differentiate $S = Ge^{-2G}$ with respect to G and set the result to 0 and solve for G, we find that the maximum occurs when $G = 0.5$, and for that $S = 1/2e = 0.18$. So, the maximum throughput is only 18% of capacity

Slotted ALOHA:

Slotted ALOHA was invented to improve the efficiency of pure ALOHA. In slotted ALOHA, time of the channel is divided into intervals called slots. The station can send a frame only at the beginning of the slot and only one frame is sent in each slot. If any station is not able to place the frame onto the channel at the beginning of the slot, it has to wait until the next time slot. There is still a possibility of collision if two stations try to send at the beginning of the same time slot.

Analysis of Slotted ALOHA

Note that the vulnerable period is now reduced in half. Using:

$$P_k(t) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

And setting $t = T_f$ and $k = 0$, we get

$$P_0(T_f) = \frac{(\lambda \cdot T_f)^0 e^{-\lambda T_f}}{0!} = e^{-G}$$

because $\lambda = \frac{G}{T_f}$. Thus, $S = G \cdot e^{-G}$

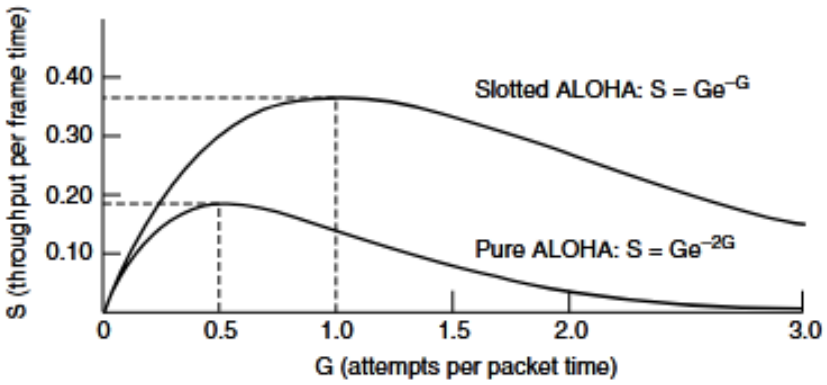


Figure 4-3. Throughput versus offered traffic for ALOHA systems.

2. Carrier Sense Multiple Access Protocols:

CSMA stands for Carrier Sense Multiple Access. Carrier Sense means, stations has an additional property with them, that they can sense the channel (carrier) and tell if the channel is in use or not. What we want, that at the start of the slot, stations should sense the channel first, and then act accordingly.CSMA was developed to overcome the problems of ALOHA i.e. to minimize the chances of collision. The chances of collision reduce to a great extent if a station checks the channel before trying to use it.

There are three different types of CSMA protocols:

1. 1-Persistent CSMA

2. Non-Persistent CSMA

3. P-Persistent CSMA

1-Persistent CSMA

In this method, station that wants to transmit data, continuously senses the channel to check whether he channel is idle or busy. If the channel is busy, station waits until it becomes idle. When the station detects an idle channel, it immediately transmits the frame. This method has the highest chance of collision because two or more stations may find channel to be idle at the same time and transmit their frames.

Non-Persistent CSMA

A station that has a frame to send senses the channel. If the channel is idle, it sends immediately. If the channel is busy, it waits a random amount of time and then senses the channel again. It reduces the chance of collision because the stations wait for a random amount of time. It is unlikely that two or more stations will wait for the same amount of time and will retransmit at the same time. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

P-Persistent CSMA

In this method, the channel has time slots such that the time slot duration is equal to or greater than the maximum propagation delay time. When a station is ready to send, it senses the channel. If the channel is busy, station waits until next slot. If the channel is idle, it transmits the frame. It reduces the chance of collision and improves the efficiency of the network.

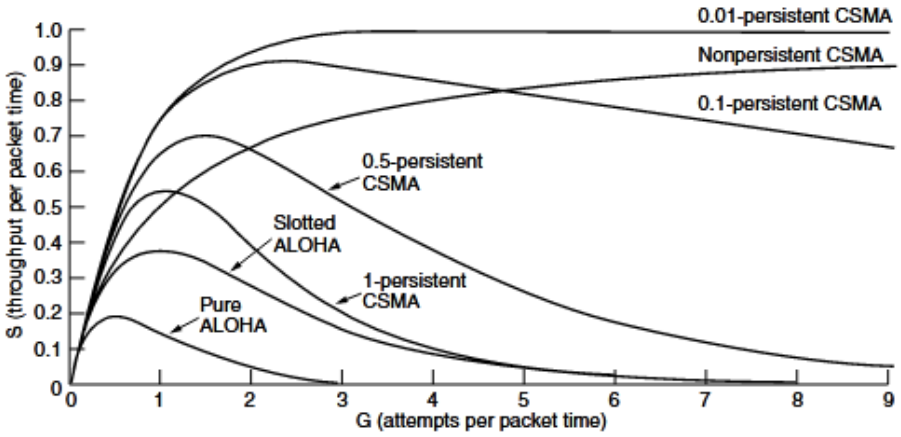


Figure 4-4. Comparison of the channel utilization versus load for various random access protocols.

CSMA with Collision Detection:

In this protocol, the station senses the channel before transmitting the frame. If the channel is busy, the station waits. Additional feature in CSMA/CD is that the stations can

detect collisions. The stations abort their transmission as soon as they detect collision. This feature is not present in CSMA. The stations continue to transmit even though they find that collision has occurred.

In CSMA/CD, the station that sends its data on the channel, continues to sense the channel even after data transmission. If collision is detected, the station aborts its transmission and waits for a random amount of time & sends its data again. As soon as a collision is detected, the transmitting stations release a jam signal. Jam signal alerts other stations. Stations are not supposed to transmit immediately after the collision has occurred.

CSMA/CD can be in one of three states: contention, transmission, or idle.

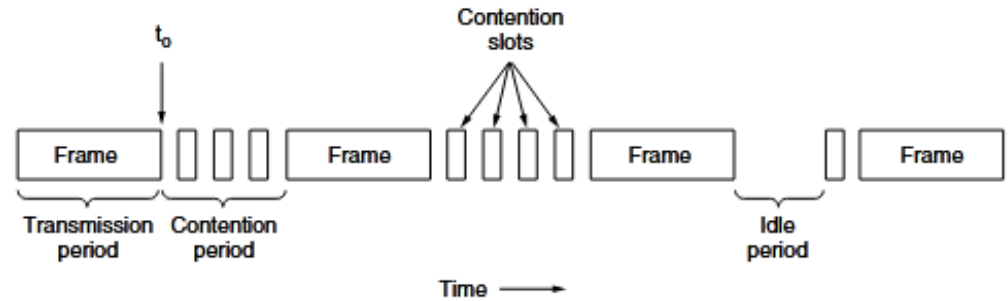


Figure 4-5. CSMA/CD can be in contention, transmission, or idle state.

Collision Free Protocols:

Almost all collisions can be avoided in **CSMA/CD** but they can still occur during the contention period. The collision during the contention period adversely affects the system performance, this happens when the cable is long and length of packet are short.

- Bit-map Protocol
- Token Passing
- Binary Countdown
- Limited-contention protocols
- Adaptive Tree Walk Protocol

Bit-map Protocol:

In our first collision-free protocol, the basic bit-map method, each contention period consists of exactly N slots. If station 0 has a frame to send, it transmits a 1 bit during the slot 0. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 bit during slot 1, but only if it has a frame queued. In general, station j may announce that it has a frame to send by inserting a 1 bit into slot j. After all N slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting frames in numerical order.

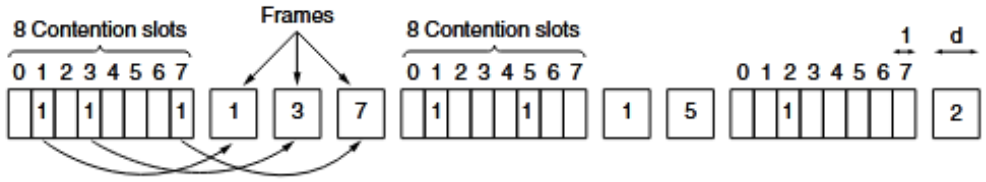


Figure 4-6. The basic bit-map protocol.

The channel efficiency at low load is easy to compute. The overhead per frame is N bits and the amount of data is d bits, for an efficiency of d /(d + N). At high load, when all the stations have something to send all the time, the N- bit contention period is prorated over N frames, yielding an overhead of only 1 bit per frame, or an efficiency of d /(d + 1). The mean delay for a frame is equal to the sum of the time it queues inside its station, plus an additional (N − 1)d + N once it gets to the head of its internal queue. This interval is how long it takes to wait for all other stations to have their turn sending a frame and another bitmap.

Token Passing:

- In token passing scheme, the stations are connected logically to each other in form of ring and access to stations is governed by tokens.
- A token is a special bit pattern or a small message, which circulate from one station to the next in some predefined order.
- In Token ring, token is passed from one station to another adjacent station in the ring whereas in case of Token bus, each station uses the bus to send the token to the next station in some predefined order.
- In both cases, token represents permission to send. If a station has a frame queued for transmission when it receives the token, it can send that frame before it passes the token to the next station. If it has no queued frame, it passes the token simply.
- After sending a frame, each station must wait for all N stations (including itself) to send the token to their neighbors and the other N – 1 stations to send a frame, if they have one.

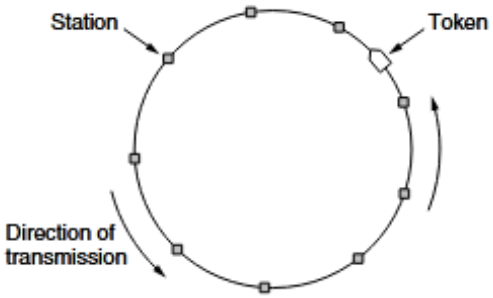


Figure 4-7. Token ring.

Binary Countdown:

Binary countdown protocol is used to overcome the overhead 1 bit per binary station. In binary countdown, binary station addresses are used. A station wanting to use the channel broadcast its address as binary bit string starting with the high order bit. All addresses are assumed of the same length.

A station wanting to use the channel now broadcasts its address as a binary bit string, starting with the high order bit. All addresses are assumed to be the same length. The bits in each address position from different stations are BOOLEAN ORed together by the channel when they are sent at the same time. We will call this protocol binary countdown.

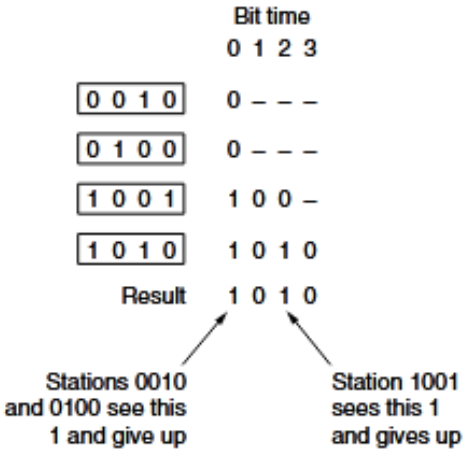


Figure 4-8. The binary countdown protocol. A dash indicates silence.

The channel efficiency of this method is d /(d + log 2 N). If, however, the frame format has been cleverly chosen so that the sender's address is the first field in the frame, even these log 2 N bits are not wasted, and the efficiency is 100%.

Limited-contention protocols:

Under conditions of light load, contention is preferable due to its low delay. As the load increases, contention becomes increasingly less attractive, because the overload associated with channel arbitration becomes greater. Just the reverse is true for contention - free protocols. At low load, they have high delay, but as the load increases, the channel efficiency improves rather than getting worse as it does for contention protocols.

Obviously it would be better if one could combine the best properties of the contention and contention - free protocols, that is, protocol which used contention at low loads to provide low delay, but used a contention-free technique at high load to provide good channel efficiency. Such protocols do exist and are called Limited contention protocols.

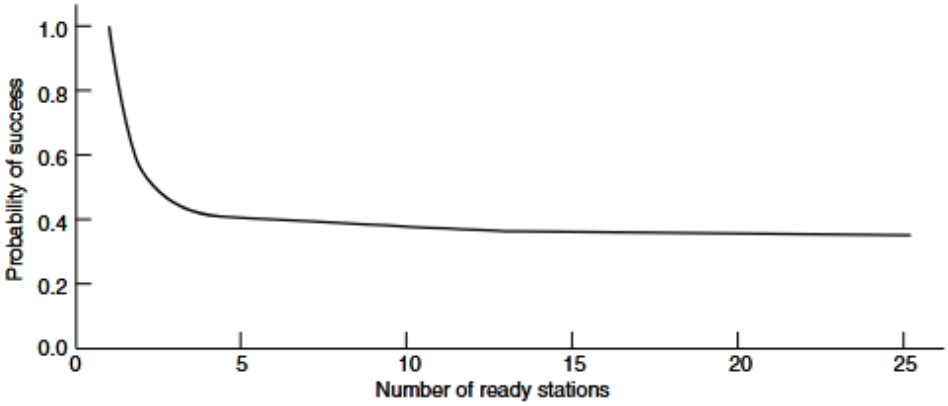
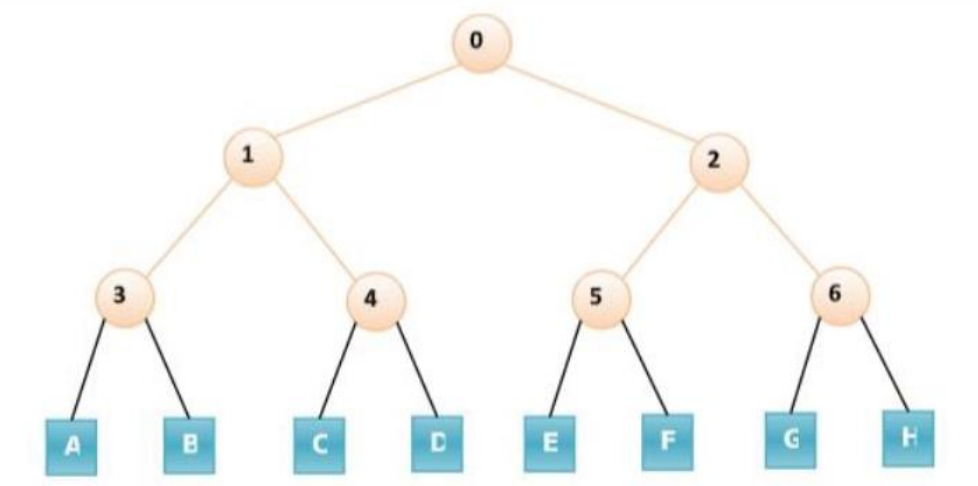


Figure 4-9. Acquisition probability for a symmetric contention channel.

It is obvious that the probability of some station acquiring the channel could only be increased by decreasing the amount of competition. The limited contention protocols do exactly that. They first divide the stations up into (not necessarily disjoint) groups. Only the members of group 0 are permitted to compete for slot 0. The competition for acquiring the slot within a group is contention based. If one of the members of that group succeeds, it acquires the channel and transmits a frame. If there is collision or no node of a particular group wants to send then the members of the next group compete for the next slot. The probability of a particular node is set to a particular value (optimum).

Adaptive Tree Walk Protocol:

In adaptive tree walk protocol, the stations or nodes are arranged in the form of a binary tree as follows –



Initially all nodes (A, B ……. G, H) are permitted to compete for the channel. If a node is successful in acquiring the channel, it transmits its frame. In case of collision, the nodes

are divided into two groups (A, B, C, D in one group and E, F, G, H in another group). Nodes belonging to only one of them are permitted for competing. This process continues until successful transmission occurs.

Adaptive Tree Walk Under conditions of light load, contention is preferable due to its low delay. As the load increases, contention becomes increasingly less attractive, because the overload associated with channel arbitration becomes greater. Just the reverse is true for contention - free protocols. At low load, they have high delay, but as the load increases, the channel efficiency improves rather than getting worse as it does for contention protocols. It is obvious that the probability of some station acquiring the channel could only be increased by decreasing the amount of competition. The limited contention protocols do exactly that.

They first divide the stations up into (not necessarily disjoint) groups. Only the members of group 0 are permitted to compete for slot 0. The competition for acquiring the slot within a group is contention based. If one of the members of that group succeeds, it acquires the channel and transmits a frame. If there is collision or no node of a particular group wants to send then the members of the next group compete for the next slot. The probability of a particular node is set to a particular value (optimum).

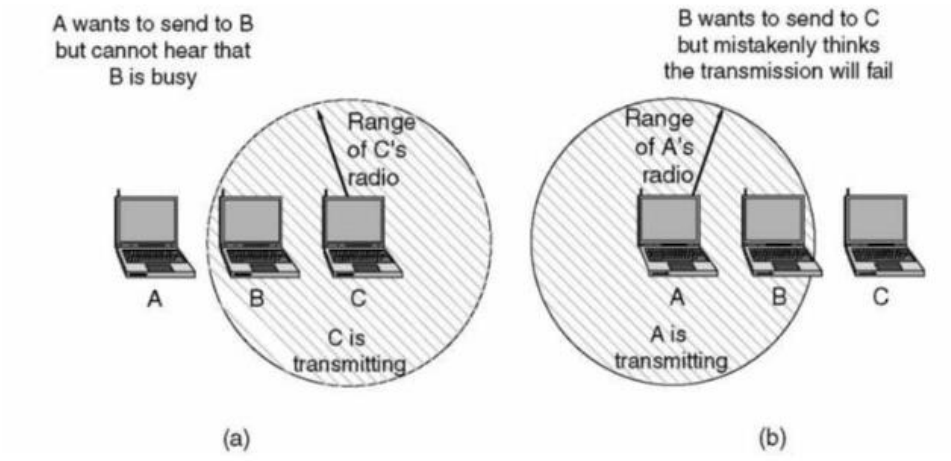
Wireless LAN Protocols:

802.11 MAC Sublayer Protocol:

Two additional problems:

– Hidden Terminal Problem

– Exposed Station Problem



.(a)The hidden station problem. (b) The exposed station problem.

MACA protocol solved hidden, exposed terminal:

– Send Ready-to-Send (RTS) and Clear-to-Send (CTS) first

– RTS, CTS helps determine who else is in range or busy (Collision avoidance).

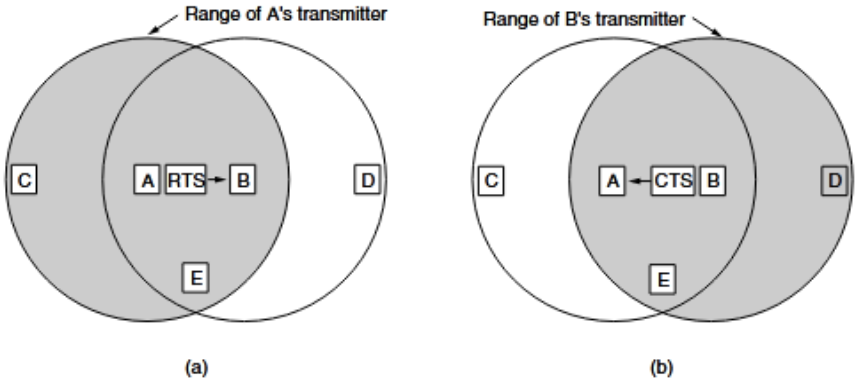


Figure 4-12. The MACA protocol. (a) *A* sending an RTS to *B*. (b) *B* responding with a CTS to *A*.

THE NETWORK LAYER

The network layer is responsible for end-to-end transmission, requiring knowledge of the network's topology and selecting appropriate paths to avoid overloading communication lines and routers. It must also address new problems when the source and destination are in different networks, primarily using the Internet and its network layer protocol, IP. This chapter will study these issues and illustrate them using the Internet.

Following are the services:

- Addressing
- Packeting
- Routing
- Internetworking

NETWORKLAYERDESIGNISSUES

1. Store-and-Forward Packet Switching

- The network layer protocols operate in a context of ISP's equipment and customers' equipment. Host H1is connected to ISP routers, A,while H2 is on LAN with leased router F,owned by the customer. These routers are considered part of the ISP network as they run the same algorithms as the ISP's routers. This context can be seen in Fig. 5-1.

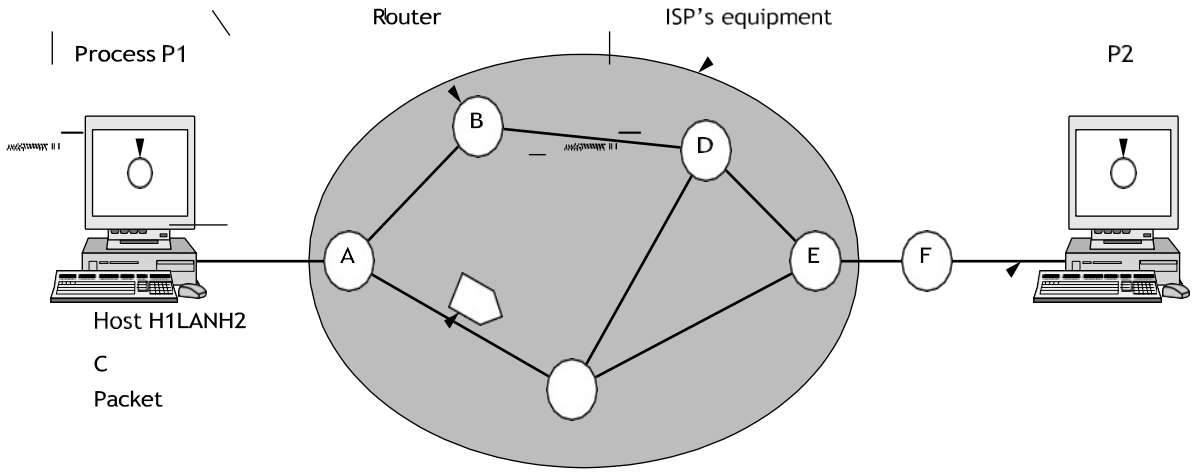


Figure5-1. The environment of the network layer protocols.

- The equipment uses store-and-forward packet switching, where a host sends a packet to the nearest router, which stores it until it's processed and verified by the link, then forwards it to the destination host.

Services Provided to the Transport Layer

The network layer provides services to the transport layer at the interface, and the services must be designed with specific goals in mind.

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a Uniform numbering plan, even across LANs and WANs.

- The network layer designers have significant freedom in creating detailed specifications for services offered to the transport layer. However, this freedom often leads to a debate between two factions: one, represented by the Internet community, who believes routers' job is to move packets and nothing else, and the other, represented by telephone companies, who believe the network should provide a reliable, connection-oriented service. This viewpoint is based on the end-to-end argument, which has been influential in shaping the Internet.

- The debate continues, with early data networks being connection oriented, but connectionless network layers have gained popularities in the early Internet days. The IP protocol is no way symbol of success, while connection-oriented technologies like ATM are now found in niche uses. However, the Internet is evolving connection-oriented features as quality of service becomes more important. Two examples of connection-oriented technologies are MPLS(Multi Protocol Label Switching) and VLANs, both widely used. The debate continues to evolve as the Internet evolves in response to the evolving needs of its users.

Implementation of Connectionless Service

- The network layer provides two types of service: connectionless and connection-oriented. Connectionless service injects packets into the network individually and routes them independently, forming datagram networks. No advance setup is needed, and The network is called a datagram network.
- On the other hand, connection-oriented service establishes a path from the source router to the destination router, forming a virtual circuit (VC) network. In a datagram network, a process P1sends a long message to a transport layer, which prepares a transport header and sends the result to the network layer. This layer operates within the operating system, similar to the physical circuits set up by telephone systems.

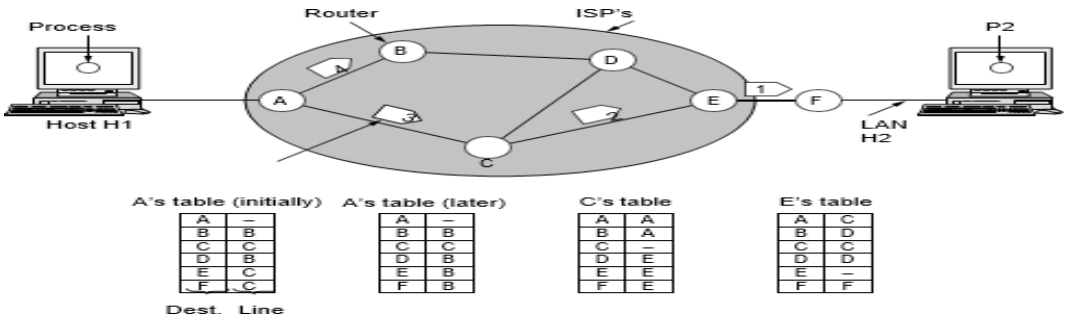


Figure 5-2. Routing within a datagram network.

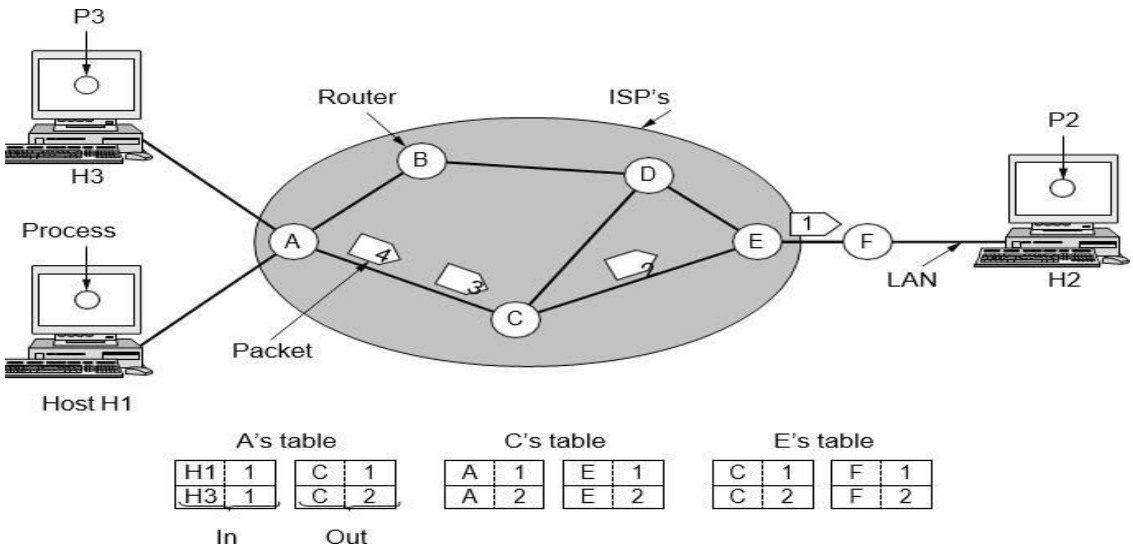
- In this example, a message is four times longer than its maximum packet size, so the network layer breaks it into four packets, 1, 2, 3, and 4, and sends each to router A using a point-to-point protocol like PPP.The ISP takes over, and every router has an internal table indicating where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.

- A router has only two outgoing lines, so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. The router's initial routing table shows that packets1, 2, and 3 are stored briefly at A,and each packet is forwarded according to A's table to C within a new frame. Packet 1 is then forwarded to E and then to F, and when it gets to F, it is sent within a frame over the LAN to H2.

- However, packet 4 is sent to router B, even though it is also destined for F. A decided to send packet 4 via a different route than the first three packets, possibly due to a traffic jam along the ACE path. The routing algorithm manages the tables and makes routing decisions, and there are several different kinds of routing algorithms.

Implementation of Connection-Oriented Service

- A connection-oriented service uses a virtual-circuit network to avoid choosing a new route for every packet sent. A route is chosen during connection setup and stored in router tables. This route is used for all traffic flowing over the connection, similar to a telephone system. Each packet carries an identification indicating its virtual circuit. For example, a packet with connection identifier 1 is sent to router C and E.



- If H3 wants to establish a connection to H2, it chooses connection identifier 1 as its only connection and tells the network to establish the virtual circuit. This leads to a conflict in the tables, as A can not distinguish connection1packets from H1 from connection1 packets from H3. To avoid conflicts, routers need the ability to replace connection identifiers in outgoing packets. This process is called label switching and is used within ISP networks in the Internet.

- An example of a connection-oriented network service is MPLS(Multi Protocol Label Switching), which is used within ISP networks in the Internet with IP packets wrapped in an MPLS header with a 20-bit connection identifier or label.MPLS is often hidden from customers, but is increasingly used to help with quality of service and other ISP traffic management tasks.

Comparison of Virtual-Circuit and Datagram Networks

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

ROUTING ALGORITHMS

The network layer primarily handles packet routing from source to destination, with most networks requiring multiple hops. Network layer design focuses on the algorithms and data structures used to choose routes and manage network segments.

- The network layer is responsible for routing packets from the source machine to the destination machine, with most networks requiring multiple hops.
- The routing algorithm is a crucial part of network layer design, responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, routing decisions must be made anew for every arriving data packet, while virtual circuits use established routes for entire sessions.
- A router has two processes: forwarding, which handles packets upon arrival, and updating the routing tables, where the routing algorithm plays a crucial role in deciding which routes to use.
- A routing algorithm should have certain desirable properties: correctness, simplicity, robustness, stability, fairness, and efficiency. Robustness is crucial for a network to cope with changes in topology and traffic without requiring all jobs to be aborted. Stability is also important, as a stable algorithm reaches equilibrium and stays there, ensuring

communication is not disrupted until the algorithm reaches equilibrium. This is especially important for major networks that may run continuously for years without system-wide failures.

- Fairness and efficiency are often contradictory goals,as seen in Fig.5-5.To maximize total flow, X to X′ traffic should be shut off, but X and X′ may not agree, requiring a compromise between global efficiency and fairness.
- Optimizing fairness and efficiency requires balancing packet delay and network throughput, often compromising by reducing packet distance or hops to improve delay and overall network throughput.

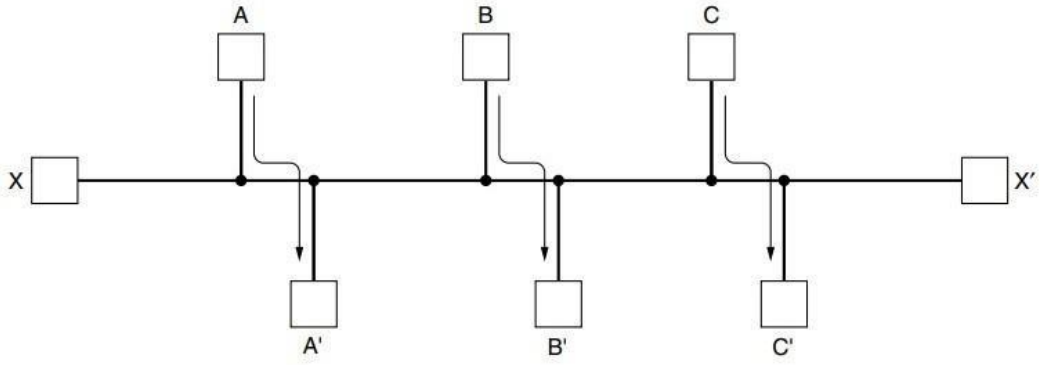


Fig:Network with a conflict between fairness and efficiency

- Static routing is a method where the choice of route is computed in advance, offline, and downloaded to routers during network booting, allowing for clear routing choices, such as sending packets to router E regardless of the final destination.
- Adaptive algorithms adapt their routing decisions to changes in topology and traffic. They differ in information source, route change frequency, and optimization metrics. These algorithms cover delivery models and can send packets to multiple, all, or one of a set of destinations. Decisions based on topology are deferred to Sec 5.3.

The Optimality Principle

- The optimality principle, as proposed by Bellman (1957), states that if router J is on the optimal path from router I to router K, the optimal path from J to K also follows the same route. This principle can be applied to specific algorithms without considering network topology or traffic.
- The optimality principle enables the creation of a sink tree,a tree rooted at the destination, where all optimal routes from all sources are used by all router.

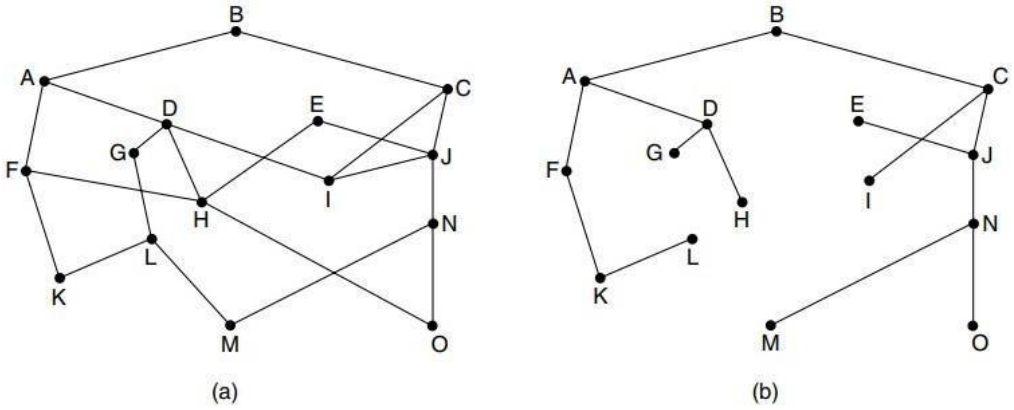


Figure 5-6. (a) A network. (b) A sink tree for router *B*.

- A sink tree is not unique, but if all possible paths are chosen, it forms a DAG (Directed Acyclic Graph) with no loops. Sink trees are used for both cases, depending on the assumption that paths do not interfere, such as a traffic jam on one path not causing another to divert.
- A sink tree is not unique, but if all possible paths are chosen, it forms a DAG (Directed Acyclic Graph) with no loops. Sink trees are used for both cases, depending on the assumption that paths do not interfere, such as a traffic jam on one path not causing another to divert.

Shortest Path Algorithm

- The concept of a shortest path deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABC and ABE in Fig. 5-7 are equally long. Another metric is the geographic distance in kilometers, in which case ABC is clearly much longer than ABE (assuming the figure is drawn to scale).

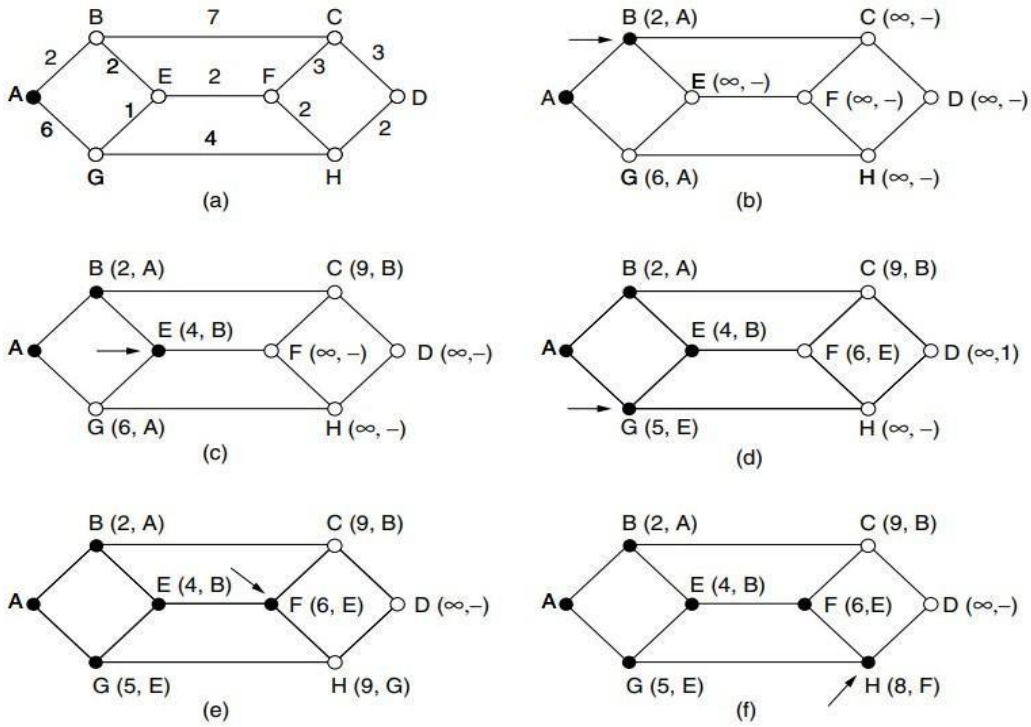


Figure 5-7. The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

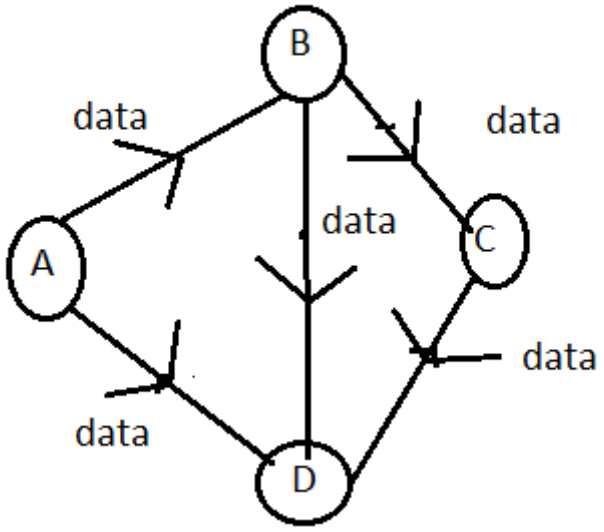
- Dijkstra's algorithm, developed in 1959, finds the shortest paths between a source and all destinations in a network. Each node is labeled with its distance from the source node along the best known path, which must be non-negative. Initially, no paths are known, so all nodes are labeled with infinity. As paths are found, labels may change to reflect better paths, and labels can be tentative or permanent.
- The labeling algorithm is a method used to find the shortest path from a node A to a node D in a network. It starts by marking node A as permanent and then examines all nodes adjacent to it, relabeling them with distance to A.
- The node with the smallest label is made the new working node. Next, all nodes adjacent to B are examined. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, a shorter path is considered. The entire graph is searched for the node with the smallest value, making it the working node for the next round.
- The algorithm consists of six steps. The algorithm works by determining the shortest path from E to AXYZE, where node Z has already been made permanent. If it has, E has already been probed, so the AXYZE path cannot be a shorter path. If Z is still tentatively labeled, if the label is greater than or equal to E, then AXYZE cannot be a shorter path than ABE. If the label is less than E, Z becomes permanent first, allowing E to be probed from Z.

- The algorithm is given in Fig. 5-8, where global variables n and dist describe the graph and are initialized before shortest path is called. The shortest paths from t to s in an undirected graph are the same as the shortest paths from s to t.

Flooding

- Routers in routing algorithms rely on local knowledge rather than the entire network. A common technique is flooding, where every incoming packet is sent on every outgoing line except the one it arrived on. This generates numerous duplicate packets, unless measures are taken. One measure is having a hop counter in each packet header, which is decremented at each hop and discarded when it reaches zero. The hop counter should be initialized to the path length.

- Flooding with a hop count can lead to an exponential number of duplicate packets. To prevent this, routers should track which packets have been flooded and avoid sending them out again. This can be achieved by placing a sequence number in each packet received from its hosts, and having a list per source router detailing which sequence numbers have been seen.



Distance Vector Routing (Bellman-Ford,Ford-Fulkerson,Adaptive,Dynamic)

1 .A router transmits its distance vector to each of its neighbors in a routing packet.

2. Each router receives and saves the most recently received distance vector from each of its neighbors.

3. A router recalculates its distance vector when:

a. It receives a distance vector from a neighbor containing different information than before.

b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

The DV calculation is based on minimizing the cost to each destination.

As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors. Once every T msec each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor. Imagine that one of these tables has just come in from neighbor X, with Xi being X's estimate of how long it takes to get to router i. If the router knows that the delay to X is m msec, it also knows that it can reach router i via X in Xi + m msec. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table. Note that the old routing table is not used in the calculation. This updating process is illustrated in Fig.9 Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K as 8, 10, 12, and 6 msec,

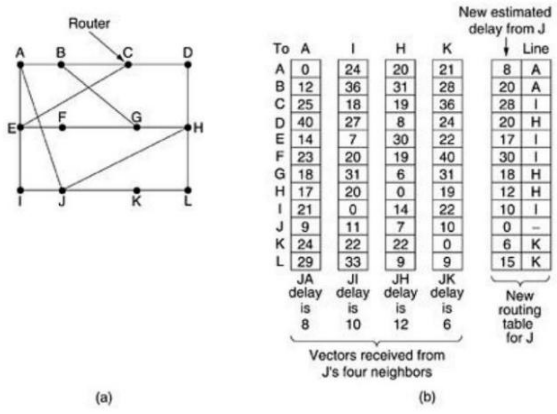


Fig.9 (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Consider how J computes its new route to router G. It knows that it can get to A in 8 msec, an A claims to be able to get to G in 18 msec, so J knows it can count on a delay of 26 msec to G it forwards packets bound for G to A. Similarly, it computes the delay to G via I, H, and K as 4 (31 + 10), 18 (6 + 12), and 37 (31 + 6) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

Example 2:

Figure 4: Subnet Diagram

To	From	From	From
A	0	2	4
B	3	0	2
C	8	4	1
D	2	7	0
E	1	11	12

CA=5 CB=6 CD=8

1 2 3

To	From	
A	5	A
B	6	B
C	0	-
D	8	D
E	6	A

Figure 5: An example of Distance Vector Routing

Part (a) shows a subnet. Part (b) the first 3 column shows the delay received from neighbor of router 'C' is A and B and D.

For example : as shown in the figure 5, 'A' claims to have 3 msec delay to B 8 msec delay to 'C' and so on.

Similarly 'B' claims to have 2 msec delay to A, 4 msec delay to 'C' and so on.

'C' has estimated his delay to neighbour A, B, D as 5, 6, 8 respectively (CA = 5, CB = 6, CD = 8).

Now (4) column shows how router 'C' decides his new route to router 'E'.

There are three ways

a) If 'C' follows line 'A' then delay is CE = CA ® AE = 5 + 1 = 6 msec.

b) If 'C' follows line 'B' then delay is CE = CB ® BE = 6 + 11 = 17 msec.

c) If 'C' follows line 'D' then delay is CD ® DE ® 8 + 12 = 20 msec.

Min delay time is via neighbor route 'A' so from C to E line is chosen 'A' in column (4) The same calculations is performed for all destination, with the new routing table as (4)

Link State Routing

- Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem). Consequently, it was replaced by an entirely new algorithm, now called link state routing. Variants of link state routing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today.

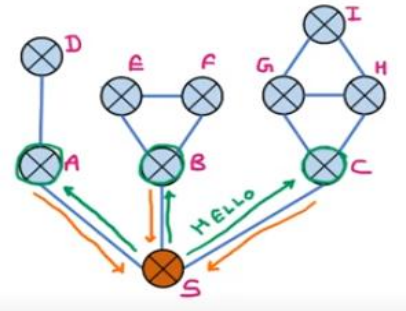
- The idea behind link state routing is fairly simple and can be stated as five parts.Each router must do the following things to make it work:

  1. Discover its neighbors and learn their network addresses.
  2. Set the distance to each of its neighbors.
  3. Construct a packet telling all it has just learned.
  4. Send this packet to and receive packets from all other routers.
  5. Compute the shortest path to every other router.

- In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router. Below we will consider each of these five steps in more detail

Learning about the Neighbors

- A router learns its neighbors by sending a special HELLO packet on each point-to-point line, and the router on the other end sends a reply with its name.



- These names must be globally unique to determine if all three routers are connected to the same S.When connected by a broadcast link, the situation becomes more

  Complicated, as shown in a broadcast LAN with three routers connected to one or more additional routers.

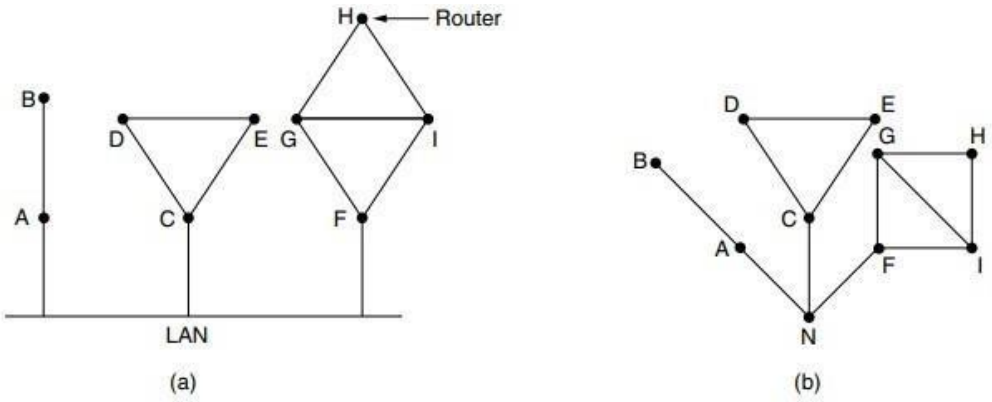
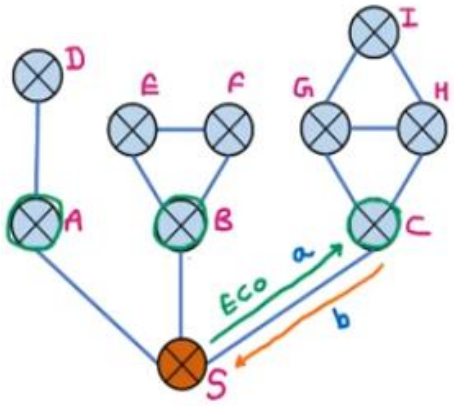


Figure 5-11. (a) Nine routers and a broadcast LAN. (b) A graph model of (a).

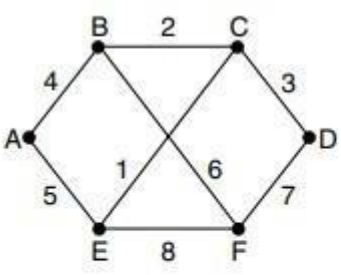
- The broadcast LAN provides connectivity between each pair of attached routers. However, modeling the LAN as many point-to-point links increases the size of the topology and leads to wasteful messages. A better way to model the LAN is to consider it as a node itself, as shown in Fig. 5-11(b). Here, we have introduced a new, artificial node, N, to which A, C, and F are connected. One designated router on the LAN is selected to play the role of N in the routing protocol. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here.
- **Setting Link Costs:** The link state routing algorithm requires each link to have a distance or cost metric for finding shortest paths. The cost to reach neighbors can be set automatically, or configured by the network operator. A common choice is to make the cost inversely proportional to the bandwidth of the link.. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is

required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.



Round trip time=a+b
Estimated Delay=(a+b)/2

- **Building Link State Packets:** Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described later) and a list of neighbors. The cost to each neighbor is also given.



(a)

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

Link State Packets

(b)

Figure 5-12. (a) A network. (b) The link state packets for this network.

- **Send this packet to and receive packets from all other routers.**
- **Compute the shortest path to every other router.**

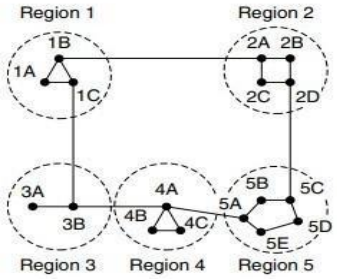
In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router. Below we will consider each of these five steps in more detail

Hierarchical Routing

- As networks grow, router routing tables grow proportionally, consuming more memory, CPU time, and bandwidth. When the network becomes too large for every router to have an entry for every other router, hierarchical routing is used.
- Routers are divided into regions, each knowing how to route packets within its own

region but not about the internal structure of other regions. This allows interconnected networks to treat each region as a separate area, allowing routers to avoid knowing the topological structure of other networks.

- For large networks, a two-level hierarchy may not be sufficient, as it may be necessary to group regions into clusters, clusters into zones, and zones into groups. For example, a packet from Berkeley to Malindi would be routed through a multilevel hierarchy.
- The Berkeley router would know the topology within California but send all out-of-state traffic to the Los Angeles router. The Los Angeles router could route traffic directly to other domestic routers but send all foreign traffic to New York. The New York router would direct all traffic to the destination country responsible for handling foreign traffic, such as Nairobi.
- The packet would then work its way down the tree in Kenya until it reached Malindi. Hierarchical routing reduces the table from 17 to 7 entries, but it also increases path length. For example, the bestroutefrom1Ato5Cisviaregion2,but all traffic to region 5 goes via region 3, as it is better for most destinations in region 5.



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

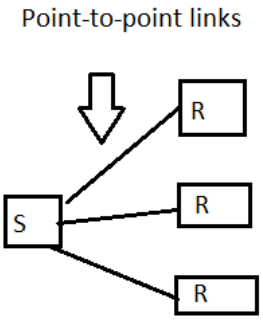
(c)

Figure 5-14. Hierarchical routing.

Broadcast Routing

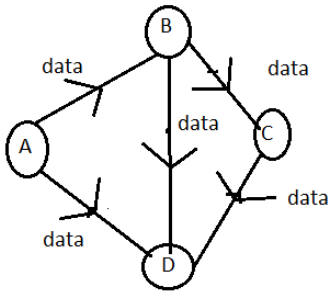
- Broadcasting is the process of sending a packet to all destinations simultaneously, such as weather reports or stock market updates.
- It can be implemented with the following techniques:

✓ Point_to_point transmission



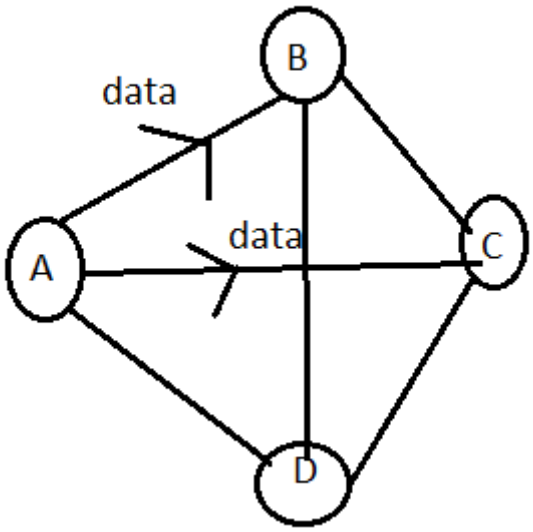
Drawback: Too many receivers need many point-to-point links.

- ✓ Flooding

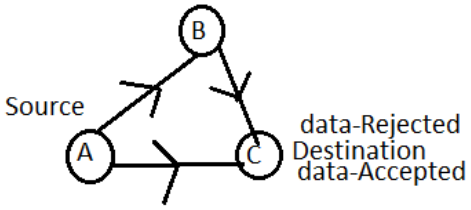


Drawback: Vast Number of duplicate Packets

- ✓ Multi Destination Routing: A sends data only to B and C not for D. Because A receives the information that destinations are only B,C



- ✓ Reverse Path Forwarding: C receives data from A directly and through B.It calculates shortest path, accepts data with shortest path and discards other. With this duplication is avoided.



✓ Spanning Tree : It establishes a tree structure that connects all devices, in the network and prevent loops .Hence Reducing network congestion .

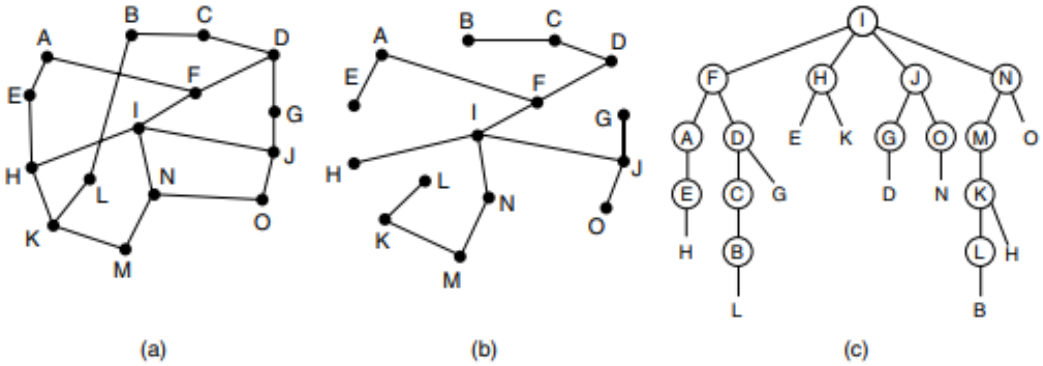


Figure 5-15. Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

Multicast Routing

- Multicasting is a method used to send messages to large but small groups in a network. This is particularly useful for applications like multiplayer games or live sports events. Sending a distinct packet to each receiver is expensive unless the group is small.
- However, broadcasting a packet is wasteful if the group consists of numerous machines on a million-node network. Multicasting requires a routing algorithm to create and destroy groups and identify routers as members. Each group is identified by a multicast address, and routers know their group membership.
- Multicast routing schemes use spanning trees to deliver packets to members of a group efficiently, based on the group's density or sparsity. Broadcast is a good start for dense groups, as it efficiently gets the packet to all parts of the network. However, it may reach routers not members of the group, which is wasteful. Deering and Cheriton (1990) proposed pruning the broadcast spanning tree by removing links that do not lead to members, resulting in an efficient multicast spanning tree.
- For example, consider two groups, 1 and 2, with routers attached to hosts belonging to one or both groups. A spanning tree for the leftmost router can be used for broadcast but is overkill for multicast. A multicast spanning tree for the leftmost router sends packets only along this tree, which is more efficient than the broadcast tree due to the number of links. Different multicast groups have different spanning trees.
- The spanning tree can be pruned using link state routing, where each router is aware of

the complete topology and can construct its own pruned tree for each sender to the group. This is an example of a link state protocol called MOSPF (Multicast OSPF).

- Distance vector routing follows a different pruning strategy, with the basic algorithm being reverse path forwarding. When a router receives a multicast message for a particulargroup,itrespondswithaPRUNEmessage,preventingtheneighborfro

sending more multicasts. This recursive pruning of the spanning tree is also possible for routers with no group members among their hosts. DVMRP (Distance Vector Multicast Routing Protocol) is an example of this method.

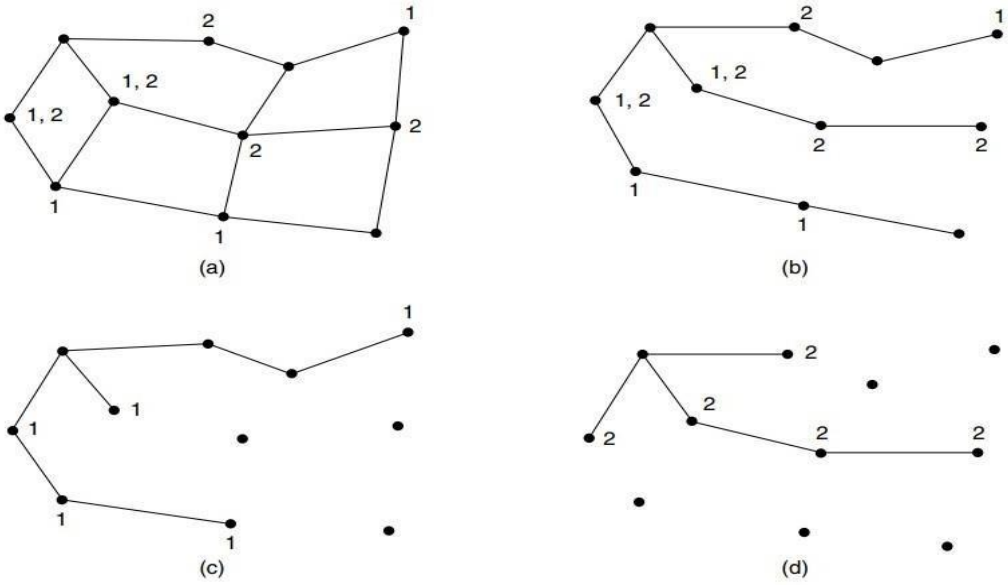


Figure 5-16. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

Anycast Routing

- Delivery models such as unicast, broadcast, and multicast are used to send packets to specific destinations. Anycast, a variant, delivers a packet to the nearest member of a group. Anycast routing schemes find these paths.
- This is useful when nodes provide services like time of day or content distribution, where the right information is received regardless of the contact node. Anycast is used in the Internet as part of DNS. Regular distance vector and link state routing can produce anycast routes.
- The distance vector routing procedure is used to assign addresses to group 1 members, resulting in nodes sending to the nearest instance of destination 1. This is because the routing protocol doesn't recognize multiple instances of destination 1, believing all instances of node 1 are the same.

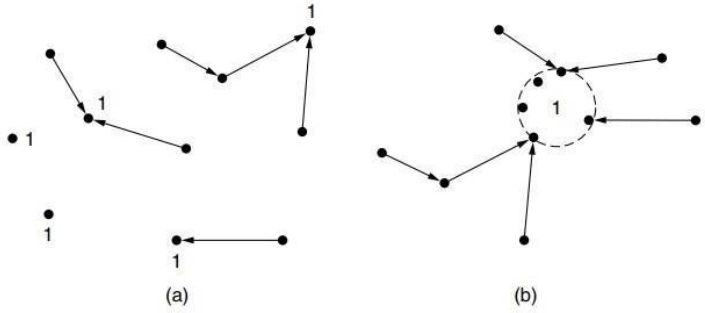
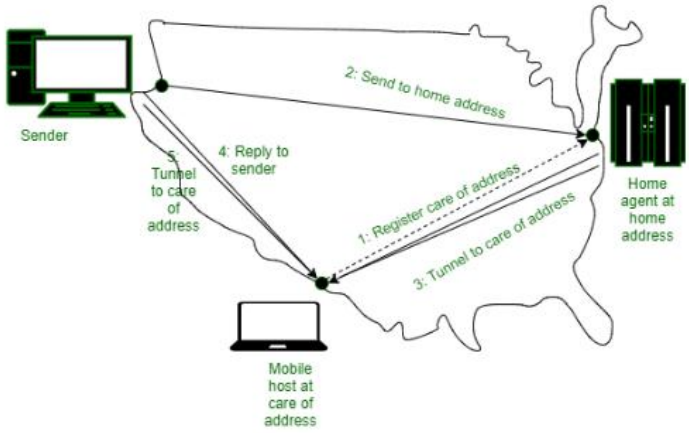


Figure 5-18. (a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

Routing for Mobile Hosts

- Millions of people use computers on the go, with mobile hosts being either stationary or mobile. As people increasingly want to stay connected, networks must find mobile hosts to route packets.
- In a model where all hosts have a permanent home location and permanent home address, the routing goal is to send packets to mobile hosts using their fixed home addresses and efficiently reach them wherever they may be. The challenge is to find mobile hosts to ensure efficient communication and reach them efficiently.
- A different model for mobile network routing could involve recompiling routes as mobile hosts move and topology changes. This would reduce the burden of constantly computing new routes. Using home addresses can also help reduce this burden. Another option is to provide mobility above the network layer, as laptops acquire new network addresses when moved to new locations.
- This model allows a laptop to browse the web but requires a higher layer location service for other hosts to send packets. Connections cannot be maintained while the host is moving, so network-layer mobility is useful.
- Mobile routing in the Internet and cellular networks involves the mobile host indicating its current location to a home agent, who forwards packets to the mobile host's home location. This process is crucial for ensuring efficient communication and delivery of

data. The mobile routing system involves a triangle routing, where the mobile host retrieves the packet from the sender and sends its reply packet directly to the sender. This route may be circuitous if the remote location is far from the home location.



The message is shown with a dashed line in the figure indicate that it is a control message, not a data message. The sender sends a data packet to the mobile host using its permanent address. This packet is routed by the network to the host home location because the home addresses belong there. It encapsulates the packet with a new header and sends this bundle to the care-of address. This mechanism is called tunneling. It is very important on the internet, so we will look at it in more detail later.

- When the encapsulated packet arrives at the care-of address, the mobile host unwraps it and retrieves the packet from the sender.
- The overall route is called triangle routing because it way is circuitous if the remote location is far from the home location.

Routing in Ad Hoc Networks

- We have now seen how to do routing when the hosts are mobile but the routers are fixed. An even more extreme case is one in which the routers themselves are mobile. Among the possibilities are emergency workers at an earthquake site, military vehicles on a battlefield, a fleet of ships at sea, or a gathering of people with laptop computers in an area lacking 802.11
- In all these cases, and others, each node communicates wirelessly and acts as both a host and a router. Networks of nodes that just happen to be near each other are called ad hoc networks or MANETs (Mobile Ad hoc NETworks). Let us now examine them briefly. More information can be found in Perkins (2001).
- What makes ad hoc networks different from wired networks is that the topology is suddenly tossed out the window. Nodes can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid barring failures, which are hopefully rare. With an ad hoc network, the topology may be changing gall the time, so the desirability and even the validity of paths can change spontaneously without warning. Needless to say, these circumstances make routing in ad hoc networks more challenging than routing in their fixed counterparts.
- Numerous routing algorithms for ad hoc networks have been proposed, but their practical use is limited compared to mobile networks. One popular algorithm is AODV (Ad hoc On-demand Distance Vector), adapted for mobile environments with limited bandwidth and battery lifetimes. It

discovers and maintains routes, highlighting its potential in ad hoc networks.

- **Route Discovery:** The Ad Hoc Network (AODV) is a network algorithm that discovers routes toadestinationondemand,savingtimeandeffortwhenthetopologychanges before the route is used. The topology of an ad hoc network can be described by a graph of connected nodes, with two nodes connected if they can communicate directly using their radios. For simplicity, all connections are symmetric.
- The algorithm is based on a newly formed ad hoc network, where a process at node A sends a packet to node I.The algorithm maintains a distance vector table at each node, keyed by destination, providing information about the destination and its neighbors. If no entry is found for node I, the process must discover a route to it. This "on demand" approach makes the algorithm efficient and efficient.

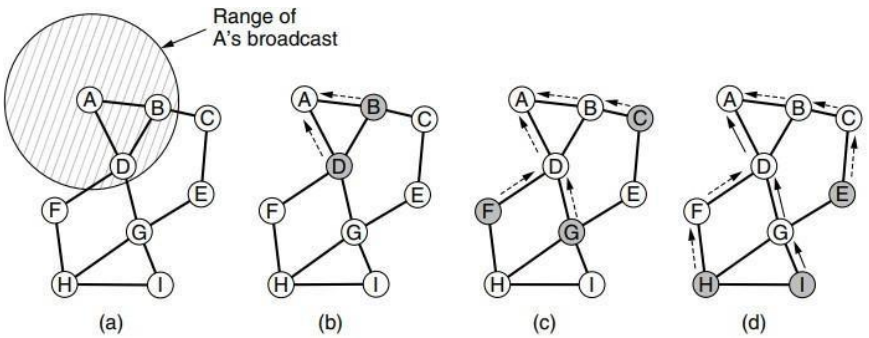


Figure 5-20. (a) Range of *A*'s broadcast. (b) After *B* and *D* receive it. (c) After *C*, *F*, and *G* receive it. (d) After *E*, *H*, and *I* receive it. The shaded nodes are new recipients. The dashed lines show possible reverse routes. The solid lines show the discovered route.

- A sends a ROUTE REQUEST packet to I, which is broadcasted using flooding. The packet is then rebroadcast to nodes F, G, C, H, E, and I. A sequence number is set at the source to remove duplicates during the flood.
- The request reaches node I, which constructs a ROUTE REPLY packet. This packet is unicast along the reverse of the path followed by the request. Intermediate nodes must remember the node that sent the request and increment a hop count as they forward the reply. The replies indicate which neighbor to use to reach the destination. Intermediate nodes G and D input the best route they hear into their routing tables. When the reply reaches A, a new route, ADGI, is created.
- The algorithm generates numerous broadcasts in large networks, even for nearby destinations. To reduce overhead, the scope of broadcasts is limited using the IP packet's Time to live field. If the field is 0, the packet is discarded. The route discovery process is modified by broadcasting a ROUTE REQUEST packet with a Time to live set to 1, and then sending more packets in increasingly wider rings, starting locally and then globally.
- **Route Maintenance:** The topology of a network can change spontaneously due to nodes moving or being switched off. The algorithm must handle this by broadcasting a Hellomessagetoitsneighbors,whichisexpectedtorespond.Ifnoresponseisreceived,the

  Broadcaster knows that the neighbor has moved out of range or failed, and if a packet is sent to a non-responsive neighbor, it learns that the neighbor is no longer available.

- This information is used to purge routes that no longer work. Each node, N, tracks its active

neighbors who have sent a packet for a possible destination during the last ΔT seconds. When any of N's neighbors becomes unreachable, it checks its routing table to see which routes have routes using the now-gone neighbor. The active neighbors inform each other recursively until all routes depending on the now-gone node are purged from all routing tables.

- Invalid routes are removed from the network, and senders can find new valid ones using the discovery mechanism. However, distance vector protocols can suffer from slow convergence or count-to-infinity problems after topology changes. To ensure rapid convergence, routes include a sequence number controlled by the destination, which increments every time a fresh ROUTE REPLY is sent. Senders request a fresh route by including the destination sequence number of the last used route in the ROUTE REQUEST.
- Intermediate nodes store routes with a higher sequence number or the few hops for the currentsequencenumber.Thison-demandprotocolsavesbandwidthandbatterylifecompared to a standard distance vector protocol that broadcasts updates periodically.
- The text discusses ad hoc routing schemes, such as DSR (Dynamic Source Routing) and GPSR (Greedy Perimeter Stateless Routing). These schemes share route discovery and maintenance when routes overlap, allowing for resource savings. For example, if B wants to send packets to I, it will perform route discovery first, then reach D, which already has a route to I. The choice of protocol depends on the types of ad hoc networks that prove useful in practice.

Congestion Control Algorithms

Too many packets present in (apart of) the network causes packet delay and loss that degrades performance. This situation is called **congestion.**

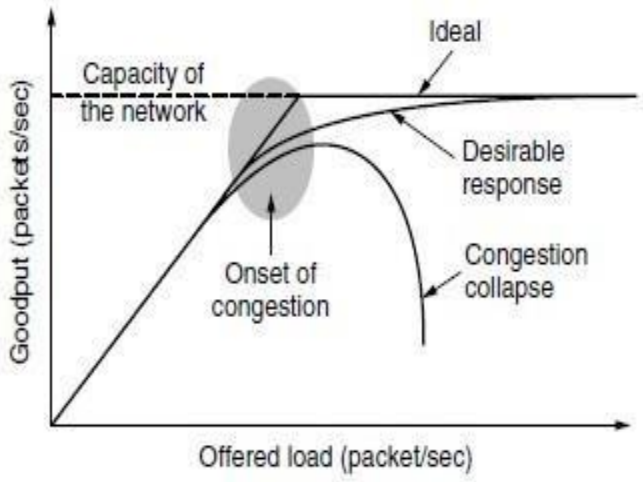


Figure3.1. With too much traffic, performance drops sharply.

- The above figure depicts the onset of congestion; the network's carrying capacity is maintained when the number of packets sent is within its limits. If twice as many packets are sent, twice as many are delivered.
- However, as traffic loads approach the capacity, some packets are lost, consuming some of the capacity, causing the number of delivered packets to fall below the ideal curve, resulting in congested networks.
- A poorly designed network can experience congestion collapse, where performance decreases as load increases beyond capacity. This occurs when packets are delayed, making them no longer useful when they leave the network.
- In the early internet, packets spent waiting for backlogs could reach their maximum time. Senders re-transmitted delayed packets, wasting capacity. To capture these factors, the y-axis of Fig. 3.1 is given as **goodput**, which is the rate at which useful packets are delivered by the network.
- Congestion is a common issue in networks, causing queues and packet loss. Adding more memory can help, but it can worsen congestion as packets time out and duplicates are sent. Low-bandwidth links or routers that process packets slowly can also become congested.
- To improve congestion, traffic can be directed away from the bottleneck, but eventually, all regions will be congested, necessitating the need for faster network design.

Differences between congestion control and flow control:

BASIS FOR COMPARISON	FLOW CONTROL	CONGESTION CONTROL
Basics	It controls the traffic from a particular sender to a receiver.	It controls the traffic entering the network.
Purpose	It prevents the receiver from being overwhelmed by the data.	It prevents the network from getting congested.
Responsibility	Flow control is the responsibility handled by data link layer and the transport layer.	Congestion Control is the responsibility handled by network layer and transport layer.
Preventive measures	The sender transmits the data slowly to the receiver.	Transport layer transmits the data into the network slowly.
Methods	Feedback-based flow control and Rate-based flow control	Provisioning, traffic-aware routing and admission control

Approaches to Congestion Control

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: **increase the resources or decrease the load.** As shown in Fig. 5-22, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

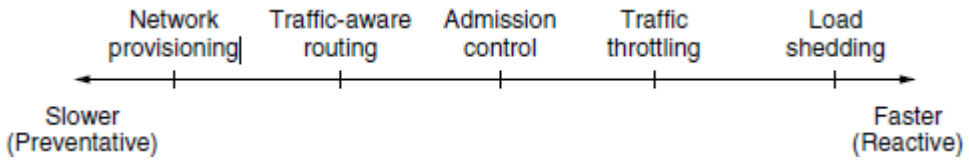


Figure3.2. Time scales of approaches to congestion control.

Network Provisioning:

- Build a network that is well matched to the traffic that it carries. If more traffic is directed but a low bandwidth link is available, definitely congestion occurs.
- Solution:

✓ Spare routers are normally used as backups.

✓ Purchasing bandwidth on an open market.

✓ Links and routers that are regularly and heavily loaded are to be upgraded.

- Sometimes resources can be added dynamically like routers and links when there is serious congestion. This is called provisioning.

Traffic-Aware Routing

Traffic awareness is one of the approaches for congestion control over the network. The basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If more traffic is directed but a low-bandwidth link is available, congestion occurs.

The main goal of traffic aware routing is to identify the best routes by considering the load, set the link weight to be a function of fixed link bandwidth and propagation delay and the variable measured load or average queuing delay.

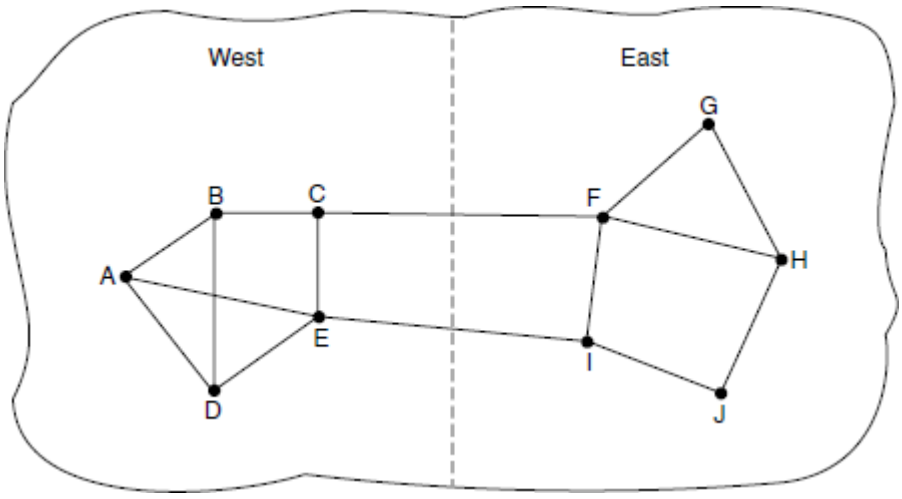


Figure3.3A network in which the East and West parts are connected by two links.

Step 1 − Consider a network which is divided into two parts, East and West both are connected by links CF and EI.

Step 2 − Suppose most of the traffic in between East and West is using link CF, and as a result CF link is heavily loaded with long delays. Including queueing delay in the weight which is used for shortest path calculation will make EI more attractive.

Step 3 − After installing the new routing tables, most of East-West traffic will now go over the EI link. As a result in the next update CF link will appear to be the shortest path.

Step 4 − As a result the routing tables may oscillate widely, leading to erratic routing and many potential problems.

Step 5 − If we consider only bandwidth and propagation delay by ignoring the load, this problem does not occur. Attempts to include load but change the weights within routing scheme to shift traffic across routes arrow range only to slow down routing oscillations.

Step 6 − the technique can contribute for successful solution, which is as follows −Multipath routing.

Admission Control:

Admission control is a technique used to prevent congestion in virtual circuit networks. The idea is to not establish a new virtual circuit if it would make the network congested. This is better than letting more circuits in when the network is already busy.

Determining when a new circuit will lead to congestion is the challenge. In telephone networks, it is straight forward since calls have fixed bandwidth. But in computer networks, virtual circuits have varying bandwidths. Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig. 3.4(a), in which two routers are congested, as indicated.

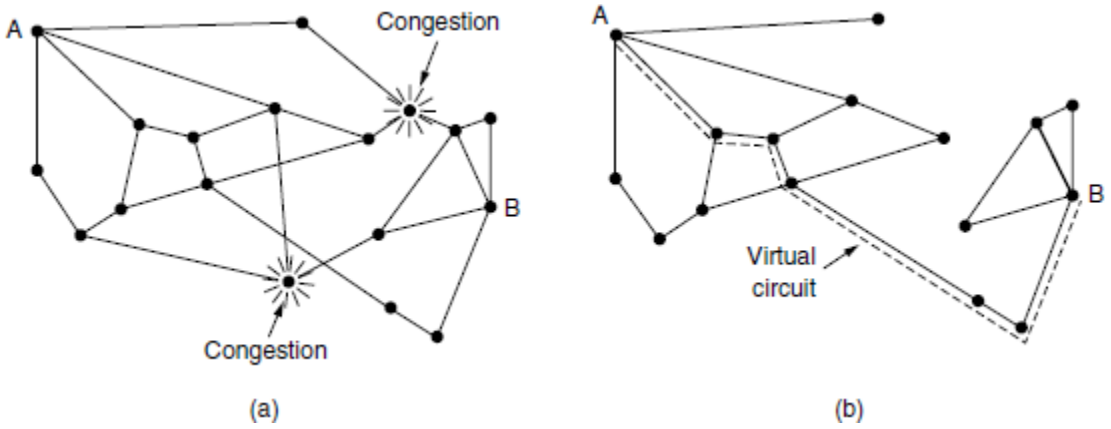


Figure3.4.(a)A congested network.(b)The portion of the network that is not congested. A virtual circuit from *A* To *B* is also shown.

Step 1 − Suppose a host attached to router A wants to set up a connection to a host attached to router B. Normally this connection passes through one of the congested routers.

Step 2 − To avoid this situation, we can redraw the network as shown in figure (b), removing the congested routers and all of their lines.

Step 3 − The dashed line indicates a possible route for the virtual circuit that avoids the congested routers.

Traffic Throttling

Traffic Throttling is an approach used to avoid congestion. In networks and the internet, the senders try to send as much traffic as possible as the network can readily deliver. In a network when congestion is approaching it should tell the senders of packets to slow down them. Traffic Throttling can be used in virtual circuit networks and datagram networks.

Let us now look at some approaches to throttling traffic that can be used in both datagram networks and virtual-circuit networks. Each approach must solve two problems.

Problem 1- The router must be able to determine when the congestion is approaching. It must identify the congestion before it has arrived. For this, each router used in the network must continuously check for all the resources and their activities in the network. Router can continuously monitor using three possibilities. They are:

- Using output links
- To buffer the most useful and priority-based packets inside the router
- The total number of packets that are lost because of insufficient buffering

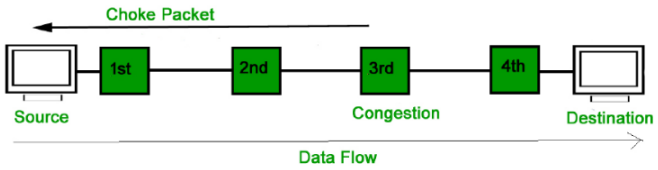
Problem 2- The second problem is that the router must send the feedback on time to the senders that are creating congestion. To deliver this feedback, the router must identify the senders properly. The router needs to send them warning efficiently without sending more packets in an already **congested** network. Different feedback mechanisms are used for solving this problem.

Feedback Mechanisms

1. Choke packets

Choke packets are a mechanism where the router directly sends the choked packet back to its sender or host. The header bit of the original packet is turned on so that it will not be able to generate any choke packet. At the time of congestion to decrease the load router will send back only the choked packets at a lower rate.In the case of datagram networks randomly, the packets are selected

therefore it leads to more choked packets. The below diagram describes the choke packets approach.



When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by *X* percent.
Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets,

- the host should ignore choke packets referring to that destination for a fixed time interval.
- After that period has expired, the host listens for more choke packets for another interval.
- If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again.
- If no choke packets arrive during the listening period, the host may increase the flow again.

2. Explicit Congestion Notification

In the explicit congestion notification approach the router does not send extra packets to the host but sets a bit of any one of the packet headers to inform that the network has approached with congestion. When any packet is delivered in the network the destination sends a reply packet to the sender informing that congestion has occurred. In the case of choke packets, the sender then throttles its transmission. The below diagram describes the explicit congestion notification.

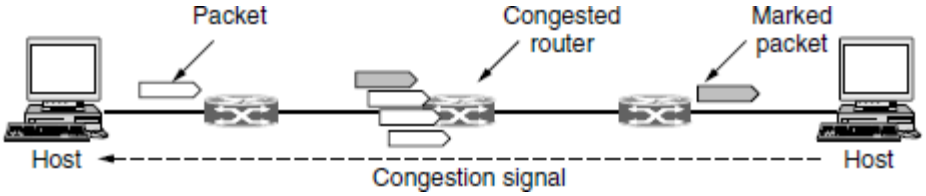
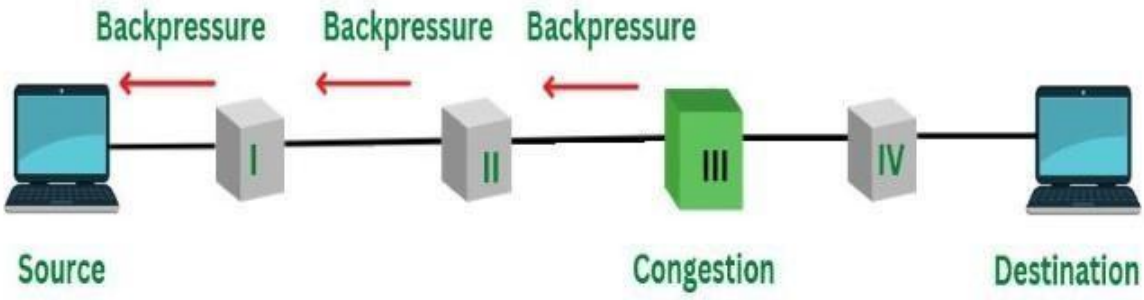
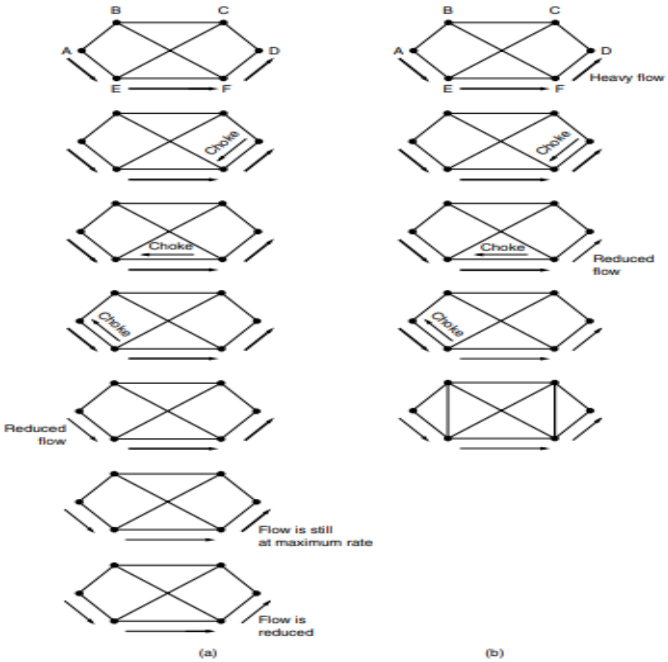


Fig3.5: Explicit congestion notification

3. Hop-by-Hop Backpressure

After the congestion has been signaled still due to a slow signal many packets are received from the long distances. The choke packets have an effect at every step and each router requires more buffers. The main aim of this Hop-by-Hop Backpressure technique is to provide faster relief at the point of congestion in the network. This technique propagates in the opposite direction of the data and is majorly used in virtual circuits. The below diagram describes Hop-by-Hop Backpressure in detail.





(a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.

Load Shedding

1. Load shedding is one of the techniques used for congestion control. A network router consists of a buffer. This buffer is used to store the packets and then route them to their destination. Load shedding is defined as an approach of discarding the packets when the buffer is full according to the strategy implemented in the data link layer. The selection of packets to discard is an important task. Many times packets with less importance and old packets are discarded.
2. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.
3. **Selection of Packets to be discarded:** In the process of load shedding the packets need to be discarded in order to avoid congestion. Therefore which packet needs to be discarded is a question. Random early detection is one of the approach used to discard the packets.

Random Early Detection

- In this approach, the router discards one or more packets before the buffer is full.
- Each time packet arrives, the RED computes average queue length.
- If avrg<threshold, do nothing else discard.
- ECN (Explicit Congestion Notification) is the preferred option if it is available. It works in exactly the same manner, but delivers a congestion signal explicitly rather than as a loss.

QUALITY OF SERVICE

- Quality of service (QoS) refers to a network's ability to achieve maximum bandwidth and deal with other network performance elements like latency, error rate and uptime.
- Quality of service also involves controlling and managing network resources by setting priorities for specific types of data (video, audio, files) on the network.
- QoS is exclusively applied to network traffic generated for video on demand, IPTV, VoIP, streaming media, videoconferencing and online gaming.

Application Requirements:

Definition of Flow:

- A stream of packets moving from a source to a destination is termed a flow.
- Flows can encompass all packets in a connection in a connection-oriented network or all packets sent from one process to another in a connectionless network.

Characterization of Flow Needs:

- Each flow's requirements can be described by four primary parameters: bandwidth, delay, jitter, and loss.
- These parameters collectively determine the Quality of Service (QoS) necessary for the flow.

Reliability:

It is one of the main characteristics that the flow needs. If there is a lack of reliability then it simply means losing any packet or losing an acknowledgement due to which retransmission is needed.

Delay:

Another characteristic of the flow is the delay in transmission between the source and destination. During audio conferencing, telephony, video conferencing, and remote conferencing there should be a minimum delay.

Jitter:

It is basically the variation in the delay for packets that belongs to the same flow. Thus Jitter is basically the variation in the packet delay.

Bandwidth:

Bits per second- The different applications need different bandwidth.

Common Applications and Network Requirements:

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Figure 5-27. Stringency of applications' quality-of-service requirements.

Techniques for Achieving Good QoS:

- Over provisioning
- Buffering
- Traffic Shaping
  - ✓ The Leaky Bucket Algorithm
  - ✓ The Token Bucket Algorithm
- Resource Reservation
- Admission Control

Over provisioning:

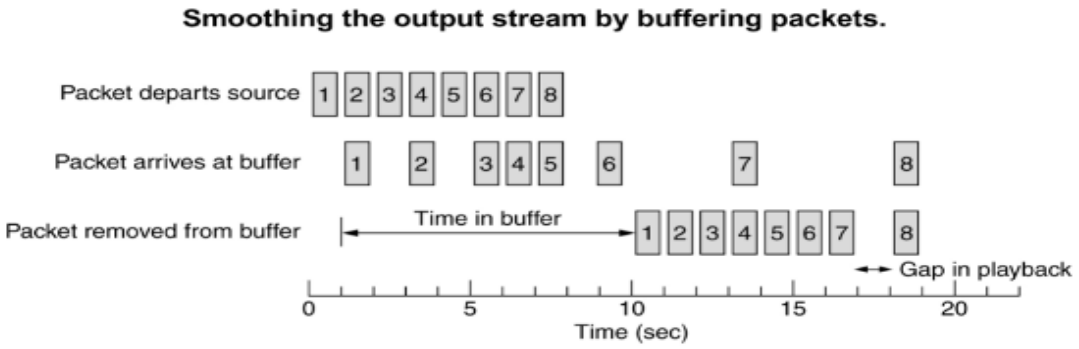
- An easy solution for ensuring good quality of service is to build a network with sufficient router capacity, buffer space and bandwidth to handle any anticipated traffic.
- This approach, known as over provisioning, allows the network to carry application traffic with minimal loss and low latency, assuming a decent routing scheme.
- The result is optimal performance, and the telephone system is an example of a somewhat over provisioned system, evident in the near-instant dial tone availability.

Drawbacks of over provisioning:

- The primary drawback of over provisioning is its high cost, essentially solving the problem by investing more resources.
- Quality of service mechanisms offers an alternative by enabling a network with less capacity to meet application requirements at a lower cost.
- Over provisioning relies on expected traffic, and significant issues may arise if traffic patterns change unexpectedly.

Buffering:

Flows can be buffered on the receiving side before being delivered. It will not affect reliability or bandwidth, but helps to smooth out jitter. This technique can be used at uniform intervals.



1. Traffic Shaping:

- Shape the traffic before it enters in to the network.
- Sender side perform the traffic shaping,i.e controls the rate at which the packets are sent.
- At the connection setup sender and carrier negotiate the traffic pattern(shape).

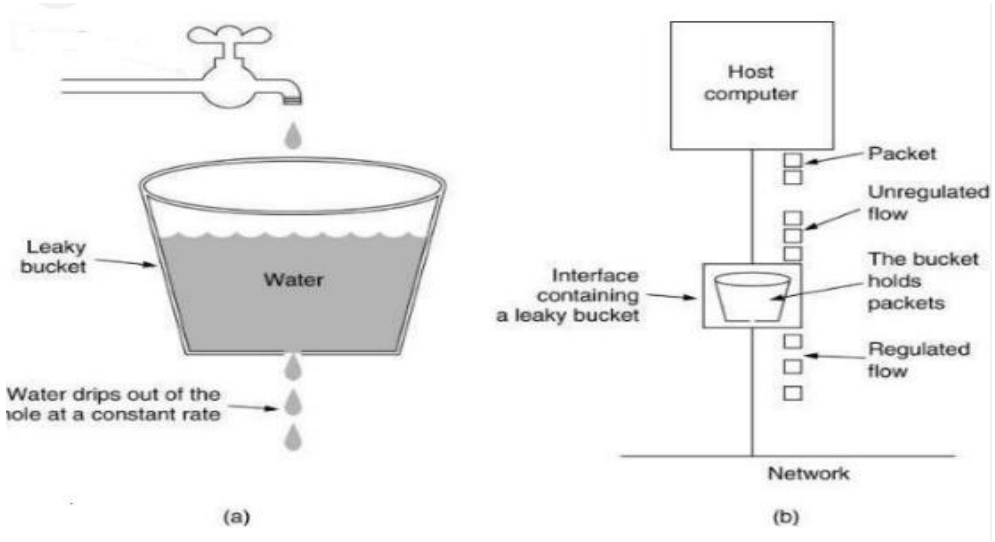
- **Traffic Characterization:**
  - Before a network can make Quality of Service(QoS) guarantees, it needs to know the nature of the traffic being guaranteed.
  - In the telephone network, the characterization is straight forward, such as a voice call needing 64 kbps with specific sampling intervals.

Two Algorithms of Traffic Shaping:

- Leaky Bucket Algorithm
- Token Bucket Algorithm

Leaky Bucket Algorithm:

Leaky bucket shaper as its name says, is based on the way a leaky bucket functions. It sends out the traffic at a fix rate even if the incoming traffic is bursty in nature. Bursty traffic could not be sent out at a time,, will be stored in the buffer (called the leaky bucket) and will be sent out once the outgoing line is free.



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

- variable-sized packets
  - allow a fixed number of bytes per tick, rather than just one packet.
  - Eg: Thus, if the rule is 1024 bytes per tick, a single 1024-byte packet can be admitted on a tick, two 512-byte packets, four 256-byte packets, and so on

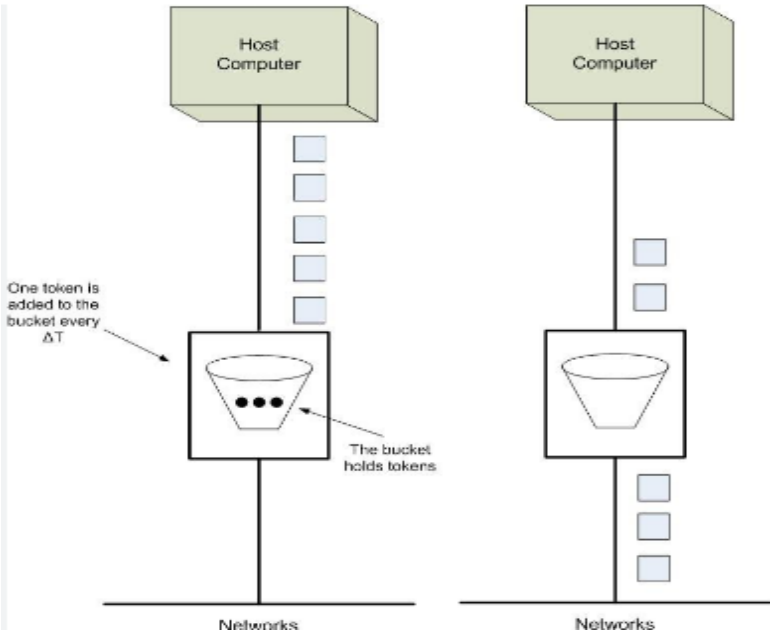
Token Bucket Algorithm:

Token bucket policy shapes the bursty traffic by allowing bursty traffic on output line but to a limit of available number of tokens.

Token bucket Policy Works as follows:

- Token are generated at regular intervals and placed into the bucket.
- The bucket has a maximum capacity of holding the tokens
- A packet can be sent to the output line only if a token is available in the bucket.
- Once a packet is sent on output line, is removed from the bucket.
- As many tokens are removed from the buckets as number of packets sent from the bucket.
- If there is no token available in the bucket, the packet cannot be sent .

  Figure below shown the working of the token bucket traffic shaping mechanism. In the figure host A sent 5 packets and there are only 3 tokens available in the bucket, hence only 3 of these packets are transmitted on the output line and 2 are hold back and will be sent once a token is placed in the bucket.



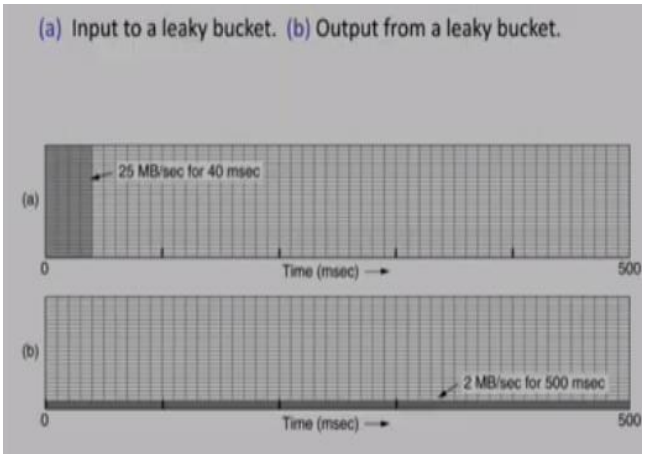
Difference between Leaky Bucket Traffic Shaper and token Bucket Traffic Shaper:

Parameter	Leaky Bucket	Token Bucket
Tokens	Token Independent	Token dependent
Overflow	If bucket overflows, then packets are discarded.	If bucket overflows, then tokens are discarded and not the packets
Data Flow Rate	Constant	Variable(Based on tokens in bucket)
Bursty traffic	Does not Supports	Supported if sufficient tokens are available.
When Application is idle	Does not do anything	Generates the tokens

Example:

Leaky Bucket

- imagine that a computer can produce data at 25 MB/sec and suppose data comes as one 40-msec burst
- However, the routers can accept this data rate only for short intervals (basically, until their buffers fill up).
- For long intervals, they work best at rates not exceeding 2 million bytes/sec(2 MB/sec)
- To reduce the average rate to 2 MB/sec, we could use a leaky bucket with ρ= 2 MB/sec

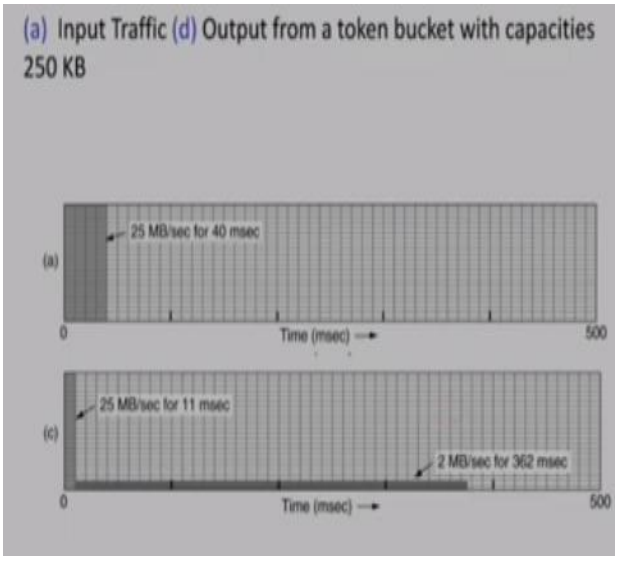


Token Bucket

- Calculating the length of the maximum rate burst is slightly tricky.
- because while the burst is being output, more tokens arrive.
- burst length S sec
- the token bucket capacity C bytes
- the token arrival rate ρ bytes/sec,
- the maximum output rate M bytes/sec
- we see that an output burst contains a maximum
  - **C + ρS bytes.**
- the number of bytes in a maximum-speed burst of length S seconds
  - **MS**
- We can solve this equation to get
  - **S = C/(M - ρ).**

- **S = C/(M - ρ).**
- For our parameters of **C = 250 KB, M = 25 MB/sec,** and **ρ=2 MB/sec**
- Substituting the values,
- **S =0.25/(25 - 2) = .011 = 11msec**
- we get a burst time of about 11 msec
- For 11msec the data can be transferred with a rate of 25MB/sec
- The remaining can be transferred with a rate of 2MB/sec

- The remaining can be transferred with a rate of 2MB/sec
- For how long?
- Inflow :25 MB/sec as one 40-msec burst
  - **$25 \times 10^6 * 40 \times 10^{-3} = 1$ MB**
  - Out of which ,the data in outflow with 25 MB/sec for 11 msec
    - **$25 \times 10^6 * 11 \times 10^{-3} = 275$ KB**
  - Remaining data (1MB- 275 KB)= 725 KB with a rate 2MB/sec
  - time= **725 K/ 2 M = 362 msec**



2. Resource Reservation (Packet Scheduling) :

Algorithms that allocate router resources among the packets of a flow and between competing flows are called packet scheduling algorithms.

Three different kinds of resources can potentially be reserved for different flows:

- Bandwidth.
- Buffer space.
- CPU cycles.

- **Bandwidth Reservation:**
  - Bandwidth is a critical resource.
  - If a flow requires 1Mbps and the outgoing line has a capacity of 2Mbps,reserving bandwidth means not oversubscribing the output line.

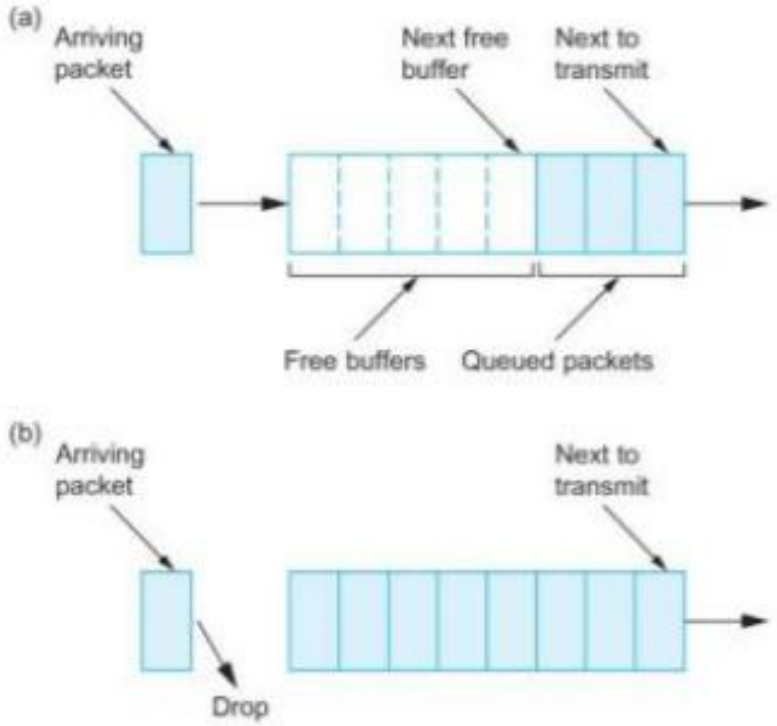
- **Buffer Space Allocation:**
  - Buffer space is another crucial resource.
  - Packets arriving at a router are buffered until they can be transmitted on the chosen outgoing line.
  - Buffers absorb small bursts of traffic as flows contend with each other.
  - For quality of service, some buffers might be reserved for specific flows, preventing competition with other flows for buffer space.
  - Maintaining a reserve ensures that a buffer is available when a flow needs it,up to a specified maximum value.
- **CPU Cycle Management:**
  - CPU cycles in routers are a potential scarce resource.
  - Processing a packet consumes router CPU time, limiting the number of packets the router can handle per second.
  - Certain packets, such as ICMP packets, may require more CPU processing.
  - Avoiding CPU overload is essential to ensure timely processing of packets, particularly those that demand additional computational resources.

Packet Scheduling Algorithms:

- FIFO (First-In First-Out)
- fair queueing
- WFQ (Weighted Fair Queueing)

1. FIFO (First-In First-Out):

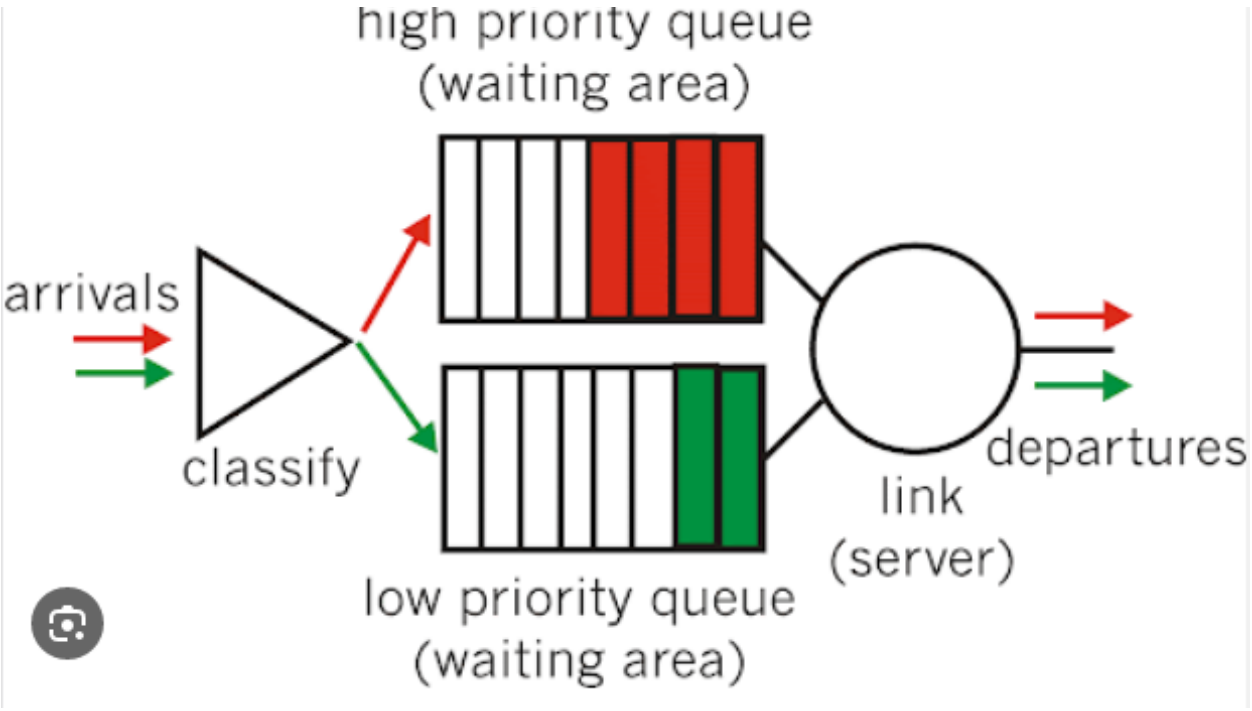
- The idea of FIFO queuing, also called first-come-first-served (FCFS) queuing, is simple:
  - The first packet that arrives at a router is the first packet to be transmitted
  - Assume the buffer space is finite. If a packet arrives and the queue is full, then the router discards that packet
  - →*Tail drop*, since packets that arrive at the tail end of the FIFO are dropped
  - Note that tail drop and FIFO are two separable ideas. FIFO is a *scheduling discipline*—it determines the order in which packets are transmitted. Tail drop is a *drop policy*—*it determines which packets get dropped*



(a) FIFO queuing; (b) tail drop at a FIFO queue.

Fair Queuing

- The main problem with FIFO queuing is that it does not discriminate between different traffic sources, or it does not separate packets according to the flow to which they belong.
- Fair queuing (FQ) maintains a separate queue for each flow currently being handled by the router.
- The router then services these queues in a sort of round-robin



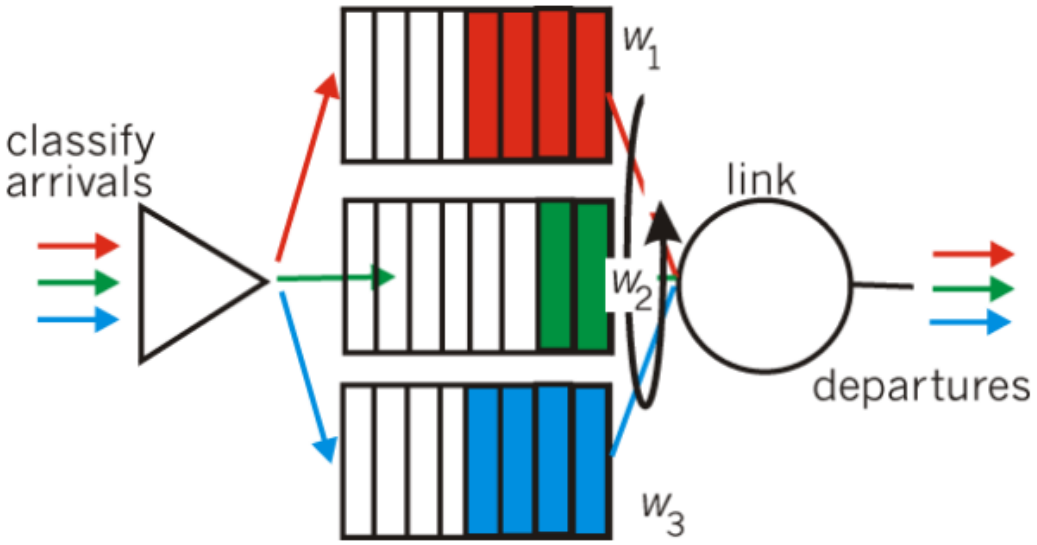


Figure 6.6-6: Weighted Fair Queuing (WFQ)

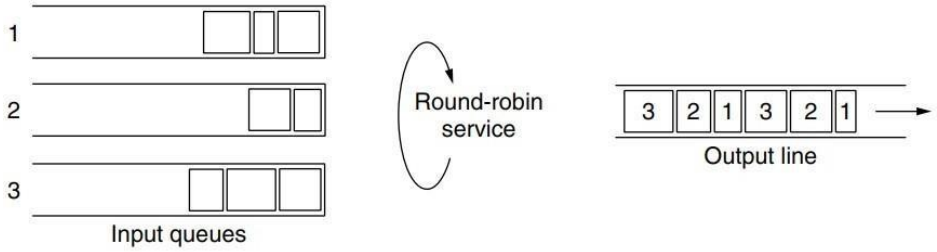
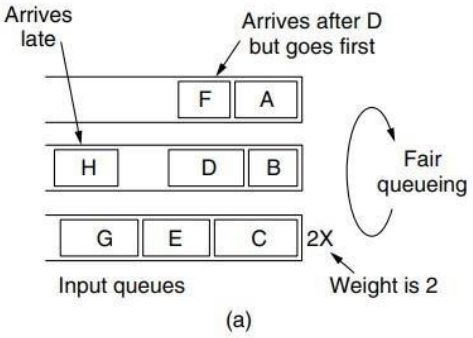


Figure 5-30. Round-robin fair queueing.



Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

(b)

Figure 5-31. (a) Weighted Fair Queueing. (b) Finishing times for the packets.

- From the table in Fig. 5-32(b), and looking only at the first two packets in the top two queues, packets arrive in the order A, B, D, and F. Packet A arrives at round 0 and is 8 bytes long, so its finish time is round 8. Similarly, the finish time for packet B is 11.Packet D arrives while B is being sent. Its finish time is 9 byte-rounds after it starts when B finishes, or 20. Similarly, the finish time for F is 16. In the absence of new arrivals, the relative sending order is A, B, F, D, even though F arrived after D. It is possible that another small packet will arrive on the top flow and obtain a finish time before D. It will only jump ahead of D if the transmission of that packet has not started
- One shortcoming of this algorithm in practice is that it gives all hosts the same priority. In many situations, it is desirable to give, for example, video servers more bandwidth than, say, file servers. This is easily possible by giving the video server two or more bytes per round. This modified algorithm is called WFQ (Weighted Fair Queueing). Letting the number of bytes per round be the weight of a flow, W, we can now give the formula for computing the finish time:

$$Fi=\max(Ai,Fi-1)+Li\ /W$$

- Where Ai is the arrival time, Fi is the finish time, and Li is the length of packet i. The bottom queue of Fig. 5-31(a) has a weight of 2, so its packets are sent more quickly as you can see in the finish times given in Fig. 5-31(b).
- As a final example of a scheduler, packets might carry timestamps and be sent in timestamp order. Clark et al. (1992) describes a design in which the timestamp records how far the packet is behind or ahead of schedule as it is sent through a sequence of routers on the path. Packets that have been queued behind other packets at a router will tend to be behind schedule, and the packets that have been serviced first will tend to be a head of schedule. Sending packets inorder of their time stamps has the beneficia leffect of speeding up slow packets while at the same time slowing down fast packets. The result is that all packets are delivered by the network with a more consistent delay.

Admission Control

- **QoS Elements and Admission Control:**
  - QoS guarantees are established through the process of admission control.
  - Admission control was initially used to control congestion, providing a weak

Performance guarantee.

- The network decides to accept or reject the flow based on its capacity and commitments to other flows.

- **Reservations and Path Considerations:**
  - Reservations must be made at all routers along the route to prevent congestion.
  - Some routing algorithms send all traffic cover the best path, potentially causing rejection of flows if there's insufficient spare capacity.
  - QoS routing and splitting traffic cover multiple paths are techniques to accommodate QoS guarantees for new flows.
- **Complexity in Acceptance or Rejection Decision:**
  - The decision to accept or reject a flow involves more than comparing requested resources with router excess capacity.
  - Applications may not have knowledge of buffers or CPU cycles, necessitating a different way to describe flows and translate descriptions to router resources.
- **Tolerance Levels and Guarantees:**
  - Some applications are more tolerant of occasional missed deadlines than others.
  - Applications must choose between hard guarantees or behavior that holds most of the time.
  - Hard guarantees are expensive due to constraining worst-case behavior, so many applications are satisfied with guarantees for most packets.
- **Negotiation of Flow Parameters:**
  - Some applications may be willing to adjust flow parameters,while others may not..
  - Properties like the number of pixels per frame and audio bandwidth may be negotiable for certain applications.
- **Importance of Accurate Flow Description:**
  - Due to the involvement of multiple parties in flow negotiation(sender,receiver, routers along the path), accurate flow description is crucial.
  - Flows need to be described in terms of specific negotiable parameters.
- **Flow Specification:**
  - A set of parameters that describe a flow is termed a flow specification.
  - The sender, such as a video server, typically generates a flow specification proposing desired parameters.
  - As the specification travels along the route, each router may modify parameters as necessary, but modifications can only reduce the flow, not increase it.
- **Parameter Modifications:**
  - Each router along the path examines and modifies the flow specification.
  - Modifications are allowed to decrease the flow, such as lowering the datarate,but

not to increase it.

- **Establishment of Parameters:**
  - When the flow specification reaches the destination, the negotiated parameters can be established.
- **Queueing Delay and Burst Size:**
  - The largest queueing delay experienced by a flow is a function of the burst size of the token bucket.
  - In cases of smooth traffic without bursts, packets are drained from the router as quickly as they arrive, resulting in no queueing delay (ignoring packetization effects).
  - Conversely, if traffic is saved up in bursts, the maximum queueing delay occurs when a maximum-size burst arrives at the router all at once. The maximum delay is the time taken to drain this burst at the guaranteed bandwidth, or B/R (ignoring packetization effects).
- **Guarantees and Token Buckets:**
  - Guarantees provided by token buckets are hard guarantees.
  - Token buckets limit the burst in essay of the traffic source.
  - Fair queueing isolates the bandwidth allocated to different flows, ensuring that each flow meets its bandwidth and delay guarantees regardless of the behavior of other competing flows at the router.
- **Isolation of Flows:**
  - Competing flows at the router cannot break the guarantee even if they save up traffic and send it all at once.
  - The combination of token buckets and fair queueing ensures that each flow's guarantees remain intact.
- **Path Through Multiple Routers:**
  - The guarantees hold for a path through multiple routers in any network topology.
  - Each flow receives a minimum bandwidth at each router, as guaranteed.
  - Maximum delay for each flow is ensured, even in the worst-case scenario where a burst of traffic hits the first router and competes with other flows. The burst's delay is at most D, and this delay smooths the burst, preventing further queueing delays at later routers.

.

Integrated and Differentiated services

- The two principle approaches to applying QoS mechanisms are Integrated Services (IntServ) and Differentiated Services (DiffServ).
- The generic name for such services are: flow-based algorithms or integrated services.
- It was aimed at both unicast and multicast applications.
- An example of unicast is a single user streaming a video clip from a news site.
- An example of multicast is a collection of digital television stations broadcasting their programs as streams of IP packets to many receivers at various locations.

IntServ

- The IntServ approach involves the reservation of bandwidth on the network from end to end between two communicating devices, before communication begins.
- This is achieved using the Resource Reservation Protocol (RSVP).
- Once the reservation is acquired, the communication takes place.
- When completed, the network resources are relinquished to be used by other applications and services.

RSVP – Resource Reservation Protocol

- **This protocol is used for making the reservations**; other protocols are used for sending the data.
- The protocol uses multicast routing using spanning trees.
- Each group is assigned a group address.
- To send to a group, a sender puts the group's address in its packets.
- The standard multicast routing algorithm then builds a spanning tree covering all group members.
- The **routing algorithm is not part of RSVP**.

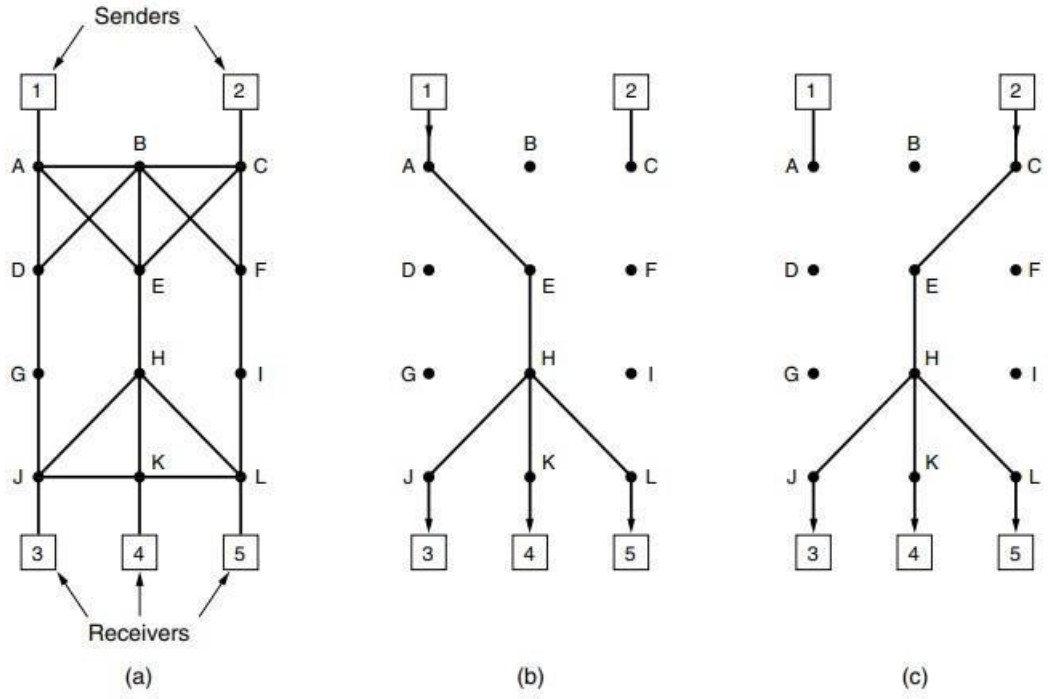


Figure 5-34. (a) A network. (b) The multicast spanning tree for host 1. (c) The multicast spanning tree for host 2.

As an example, consider the network of Fig. 5-34(a). Hosts 1 and 2 are multicast senders, and hosts 3, 4, and 5 are multicast receivers. In this example, the senders and receivers are disjoint,but in general, the two sets may overlap.The multicast trees for hosts 1 and 2 are shown in Fig. 5-34(b) and Fig. 5-34(c), respectively.

- Any of the receivers in a group can send a reservation message up the tree to the sender.
- The message is propagated using the reverse path forwarding algorithm.
- At each hop, the router notes the reservation and reserves the necessary bandwidth
- If insufficient bandwidth is available, it reports back failure.
- By the time the message gets back to the source, bandwidth has been reserved all the way from the sender to the receiver making the reservation request along the spanning tree.

- Here host 3 has requested a channel to host 1
- Once it has been established, packets can flow from 1 to 3 without congestion.
- If host 3 next reserves a channel to the other sender, host 2, so the user can watch two television programs at once.

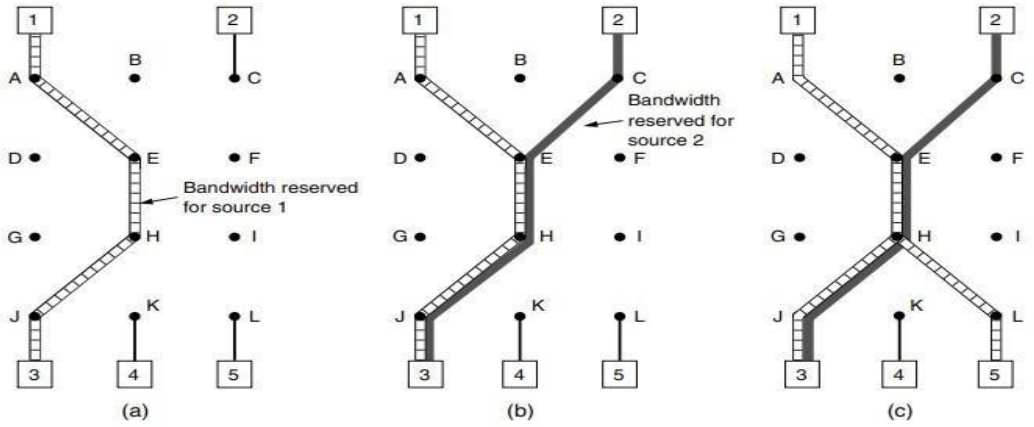


Figure 5-35. (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

- Host 5 decides to watch the program being transmitted by host 1 and also makes a reservation.
- First, dedicated bandwidth is reserved as far as router H, later router sees that it already has a feed from host 1, so if the necessary bandwidth has already been reserved, it does not have to reserve any more.
- Note that hosts 3 and 5 might have asked for different amounts of bandwidth (e.g., 3 has a small screen and 5 a bigger one with higher resolution).
- The capacity reserved must be large enough to satisfy the greediest receiver.

Drawbacks of IntServ

- All routers within a network must be configured to support and respond to RSVP requests, if only one fails, the reservation will fail.
- There may potentially be hundreds or even thousands of individual reservation requests for routers to create, maintain and tear down reservations on demand, a process that can quickly overwhelm the CPU and memory of the RSVP-enabled routers.
- Because RSVP will reserve the bandwidth from end to end, no other service can use that bandwidth, even under conditions of extreme congestion.

Class-Based Quality of Service (Differentiated Services-DiffServ):

- DiffServ is a QoS approach involving packet prioritization.
- With DiffServ, individual packets are marked according to the type of prioritization they require.
- Routers and switches use various queueing strategies to conform to these requirements.
- This approach is known as **class-based** (as opposed to flow-based) quality of service.

Differentiated service goal is to divide the traffic into classes

- Differentiated services (DS) can be offered by a set of routers forming an administrative domain e.g ISP
- The administration defines a **set of service classes** with corresponding **forwarding rules.**
- If a customer signs up for DS, customer packets entering the domain may carry a Type of Service field in them, with better service provided to some classes (e.g., premium service) than to others.
- This scheme requires no advance setup, no resource reservation, and no time-consuming end-to-end negotiation for each flow, as with integrated services.
- This makes DS relatively easy to implement.

Types of forwarding

- **Expedited forwarding**
- **Assured Forwarding**

Expedited Forwarding:

The vast majority of the traffic is expected to be regular, but a limited fraction of the packets are expedited.

- The expedited packets should be able to transit the network as though no other packets were present.
- In this way they will get low loss, low delay and low jitter service—just what is needed for VoIP.
- Packets are classified as expedited or regular and marked accordingly.
- This step might be done on the sending host or in the ingress (first) router.
- If the marking is done by the host, the ingress router is likely to police the traffic to make sure that customers are not sending more expedited traffic than they have paid for.
- Within the network, the routers may have two output queues for each outgoing line, one for expedited packets and one for regular packets.
- When a packet arrives, it is queued accordingly.
- The expedited queue is given priority over the regular one, for example, by using a priority scheduler.
- In this way, expedited packets see an unloaded network, even when there is, in fact, a heavy load of regular traffic.

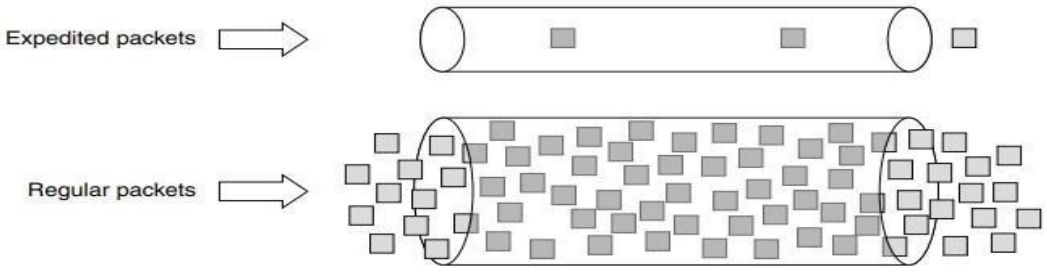


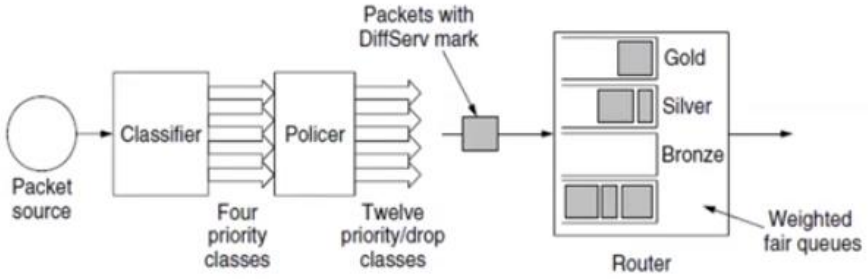
Figure 5-36. Expedited packets experience a traffic-free network.

Assured Forwarding Service Class:

- Assured forwarding specifies that there shall be four priority classes, each class having its own resources.
- The top three classes might be called gold, silver, and bronze.
- In addition, it defines three discard classes for packets that are experiencing congestion: low, medium, and high.
- Taken together, these two factors define 12 service classes.

	Low Drop Probability within class	Medium Drop Probability within class	High Drop Probability within class
Class 1	AF11	AF12	AF13
Class 2	AF21	AF22	AF23
Class 3	AF31	AF32	AF33
Class 4	AF41	AF42	AF43

Implementation of assured forwarding



- The first step is to classify the packets into one of the four priority classes.
- The next step is to determine the discard class for each packet.
- Finally, the packets are processed by routers in the network with a packet scheduler that distinguishes the different classes.

UNIT IV
TRANSPORT LAYER

Transport layer - Services - Berkeley Sockets -Example – Elements of Transport protocols – Addressing - Connection Establishment - Connection Release - Flow Control and Buffering – Multiplexing – Congestion Control - Bandwidth Allocation - Regulating the Sending Rate –UDP-RPC – TCP - TCP Segment Header - Connection Establishment - Connection Release - TCP Congestion Control

Introduction:

The network layer provides end-to-end packet delivery using data-grams or virtual circuits. The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use. It provides the abstractions that applications need to use the network.

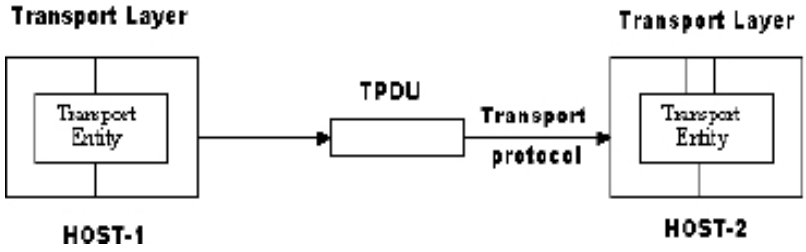
Transport Entity: The hardware and/or software which make use of services provided by the network layer, (within the transport layer) is called transport entity.

Transport Service Provider: Layers 1 to 4 are called Transport Service Provider.

Transport Service User: The upper layers i.e., layers 5 to 7 are called Transport Service User.

Transport Service Primitives: Which allow transport users (application programs) to access the transport service.

TPDU (Transport Protocol Data Unit): Transmissions of message between 2 transport entities are carried out by TPDU. The transport entity carries out the transport service primitives by blocking the caller and sending a packet the service. Encapsulated in the payload of this packet is a transport layer message for the server's transport entity. The task of the transport layer is to provide reliable, cost-effective data transport from the source machine to the destination machine, independent of physical network or networks currently in use.



TRANSPORT SERVICE

1. Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, **reliable, and cost-effective data transmission** service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the **services pro-vided by the network layer.** The software and/or hardware within the transport layer that does the work is called the **transport entity.** The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.

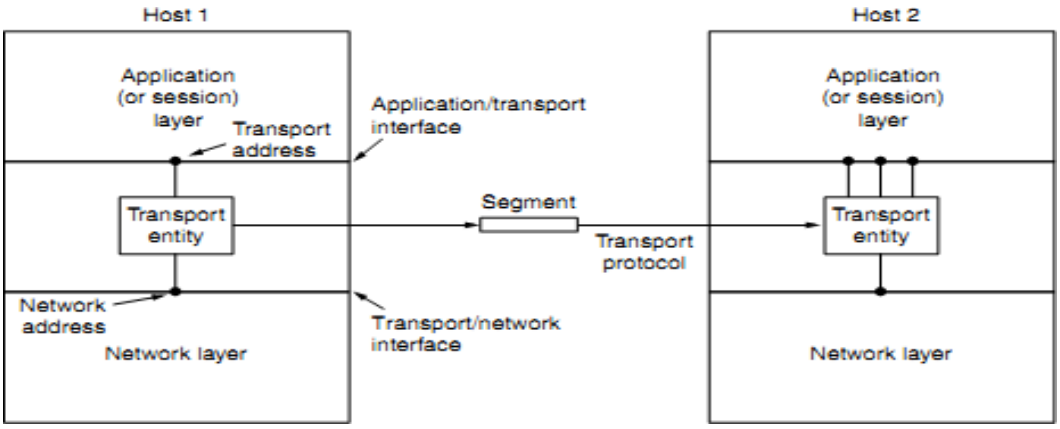


Fig 4.1: The network, Application and transport layer

There are two types of network service

- Connection-oriented
- Connectionless

Similarly, there are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways.

In both cases, connections have three phases:

- Establishment
- Data transfer
- Release.

- Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service.
- The bottom four layers can be seen as the transport service provider, whereas the upper layer(s) are the transport service user.

2. Transport Service Primitives

- To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.
- The transport service is similar to the network service, but there are also some important differences.
- The **main difference** is that the network service is intended to model the service offered by real networks. Real networks can lose packets, so the network service is generally **unreliable.**
- The (connection-oriented) transport service, in contrast, is **reliable**

As an example, consider two processes connected by pipes in UNIX. They assume the connection between them is perfect. They do not want to know about acknowledgements, lost packets, congestion, or anything like that. What they want is a 100 percent reliable connection. Process A puts data into one end of the pipe, and process B takes it out of the other.

A **second difference** between the network service and transport service is **whom the services are intended for**. The network service is used only by the transport entities. Consequently, the transport service must be convenient and easy to use.

Table:4.1 - The primitives for a simple transport service.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Eg: Consider an application with a server and a number of remote clients.

1. The server executes a “LISTEN” primitive by calling a library procedure that makes a System call to block the server until a client turns up.
2. When a client wants to talk to the server, it executes a “CONNECT” primitive, with “CONNECTION REQUEST” TPDU sent to the server.
3. When it arrives, the TE unblocks the server and sends a “CONNECTION ACCEPTED” TPDU back to the client.
4. When it arrives, the client is unblocked and the connection is established. Data can now be exchanged using “SEND” and “RECEIVE” primitives.
5. When a connection is no longer needed, it must be released to free up table space within the 2 transport entries, which is done with “DISCONNECT” primitive by sending “DISCONNECTION REQUEST”

TPDU. This disconnection can b done either by asymmetric variant (connection is released, depending on other one) or by symmetric variant (connection is released, independent of other one).

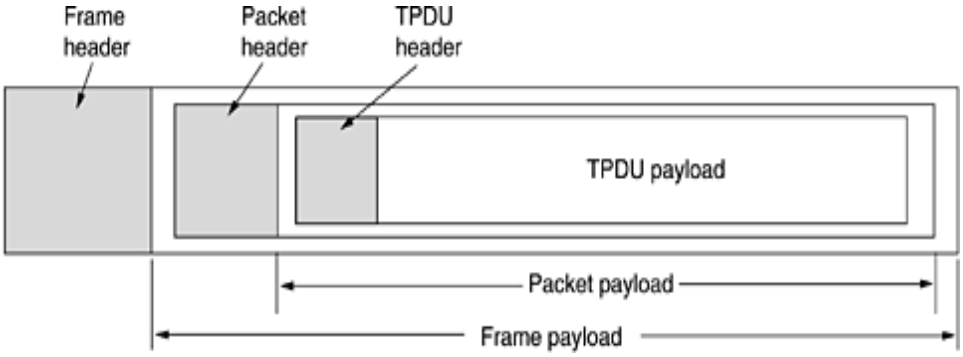


Figure 4.2 - Nesting of TPDUs, packets, and frames

- The term segment for messages sent from transport entity to transport entity.
- TCP, UDP and other Internet protocols use this term. Segments (exchanged by the transport layer) are contained in packets (exchanged by the network layer).
- These packets are contained in frames(exchanged by the data link layer).When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local delivery, passes the contents of the frame payload field up to the network entity.
- The network entity similarly processes the packet header and then passes the contents of the packet payload up to the transport entity. This nesting is illustrated in Fig. 4.2.

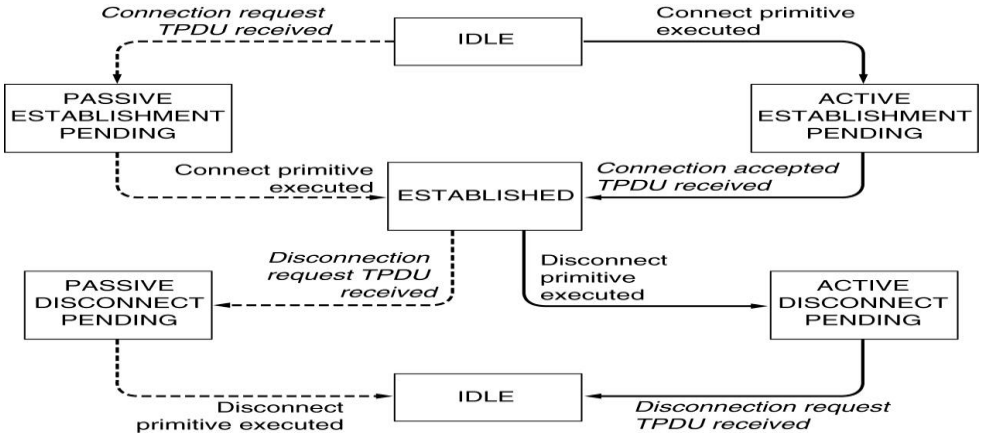


Figure 4.3 - A state diagram for a simple connection management scheme. Transitions labelled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

In fig. 4.3 each transition is triggered by some event, either a primitive executed by the local transport user or an incoming packet. For simplicity, we assume here that each TPDU is separately acknowledged. We also assume that a symmetric disconnection model is used, with the client going first. Please note that this model is quite unsophisticated. We will look at more realistic models later on.

BERKLEY SOCKETS

These primitives are socket primitives used in Berkley UNIX for TCP.

The socket primitives are mainly used for TCP. These sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular. The primitives are now widely used for Internet programming on many operating systems, especially UNIX -based systems, and there is a socket-style API for Windows called ‘‘**winsock**.’’

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 4.4 - The socket primitives for TCP.

The first four primitives in the list are executed in that order by servers.

The **SOCKET** primitive creates a new endpoint and allocates table space for it within the transport entity. The parameter includes the addressing format to be used, the type of service desired and the protocol. Newly created sockets do not have network addresses.

- The **BIND** primitive is used to connect the newly created sockets to an address. Once a server has bound an address to a socket, remote clients can connect to it.
- The **LISTEN** call, which allocates space to queue incoming calls for the case that several clients try to connect at the same time.
- The server executes an **ACCEPT** primitive to block waiting for an incoming connection.

Some of the client side primitives are. Here, too, a socket must first be created

- The **CONNECT** primitive blocks the caller and actively starts the connection process. When it completes, the client process is unblocked and the connection is established.
- Both sides can now use **SEND** and **RECEIVE** to transmit and receive data over the full-duplex connection.
- Connection release with sockets is symmetric. When both sides have executed a **CLOSE** primitive, the connection is released.

ELEMENTS OF TRANSPORT PROTOCOLS

The transport service is implemented by a transport protocol used between the two transport entities. The transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues. The difference transport protocol and data link protocol depends upon the environment in which they are operated.

These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig.

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.

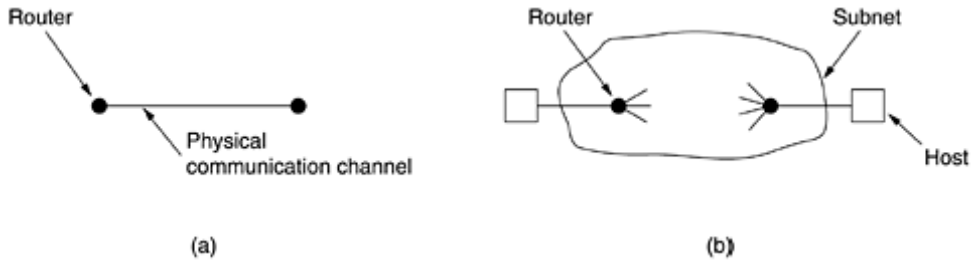


Figure (a) Environment of the data link layer. (b) Environment of the transport layer.

In the data link layer, it is not necessary for a router to specify which router it wants to talk to. In the transport layer, explicit addressing of destinations is required.

In the transport layer, initial connection establishment is more complicated, as we will see. Difference between the data link layer and the transport layer is the potential existence of storage capacity in the subnet

Buffering and flow control are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer.

The transport service is implemented by a transport protocol between the 2 transport entities.

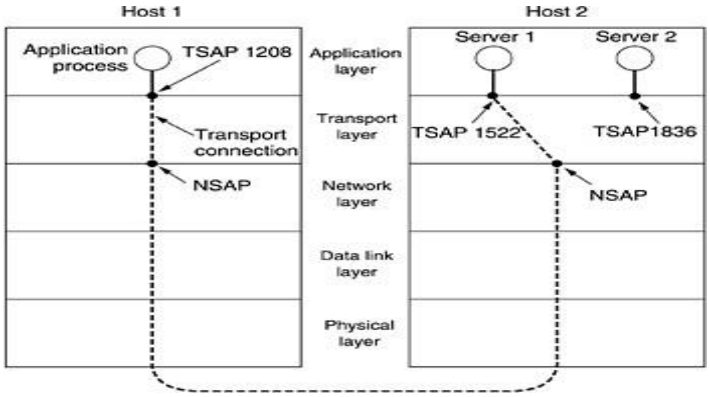


Figure 4.5 illustrates the relationship between the NSAP, TSAP and transport connection. Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP.

These connections run through NSAPs on each host, as shown. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport end points that share that NSAP.

The elements of transport protocols are:

1. ADDRESSING
2. Connection Establishment.
3. Connection Release.
4. Error control and flow control
5. Multiplexing.

1. ADDRESSING

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**.

There are two types of access points.

TSAP (Transport Service Access Point) to mean a specific endpoint in the transport layer.

The analogous endpoints in the network layer (i.e., network layer addresses) are not surprisingly called **NSAPs (Network Service Access Points).** IP addresses are examples of NSAPs.

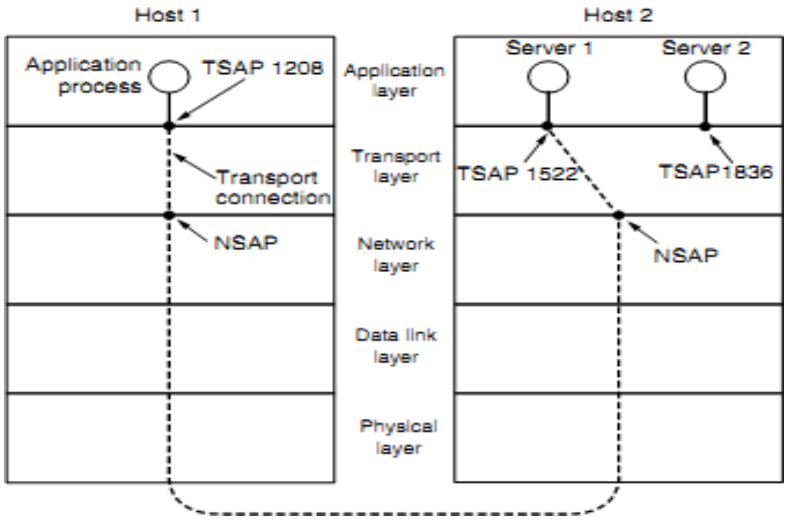


Fig 4.5: TSAP and NSAP network connections

Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport endpoints that share that NSAP.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.
3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

2. CONNECTION ESTABLISHMENT:

With packet lifetimes bounded, it is possible to devise a fool proof way to establish connections safely. Packet lifetime can be bounded to a known maximum using one of the following techniques:

- Restricted subnet design
- Putting a hop counter in each packet
- Time stamping in each packet

Using a 3-way hand shake, a connection can be established. This establishment protocol doesn't require both sides to begin sending with the same sequence number.

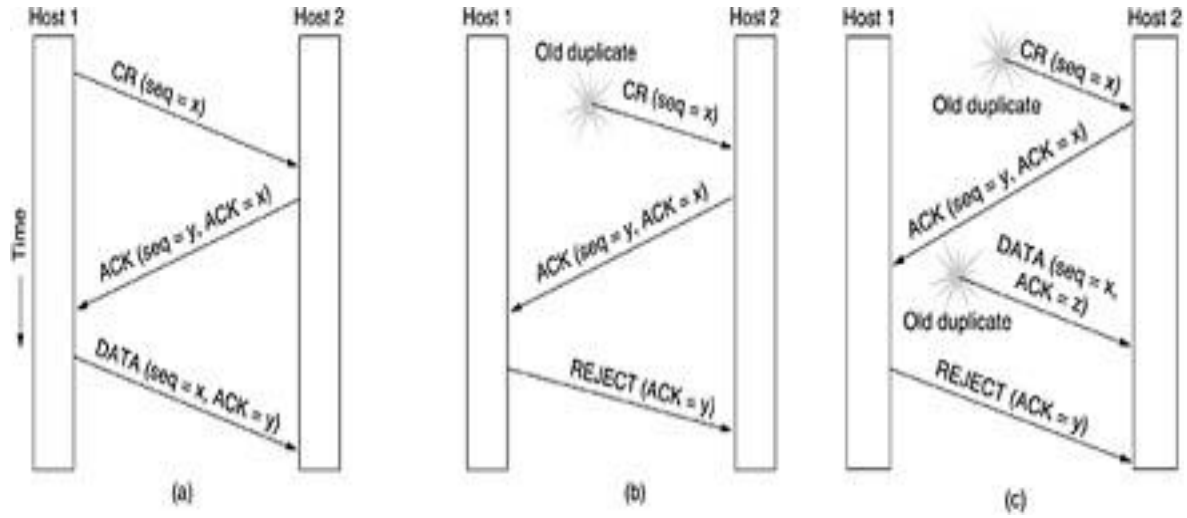


Fig 4.6: Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNEC TION REQUEST (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK .

- The **first technique** includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the longest possible path. It is difficult, given that internets may range from a single city to international in scope.

- The **second method** consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.
- The **third method** requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time.

In **fig (A)** Tomlinson (1975) introduced the **three-way handshake**.

- This establishment protocol involves one peer checking with the other that the connection request is indeed current. Host 1 chooses a sequence number, x , and sends a CONNECTION REQUEST segment containing it to host 2. Host 2replies with an ACK segment acknowledging x and announcing its own initial sequence number, y.

- Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data segment that it sends

In **fig (B)** the first segment is a delayed duplicate CONNECTION REQUEST from an old connection.

- This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host1an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection.
- When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

- The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet.

In **fig (C)** previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it.

- At this point, it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no segments containing sequence number y or acknowledgements to y are still in existence.
- When the second delayed segment arrives at host 2, the fact that z has been acknowledged rather than y tells host 2 that this, too, is an old duplicate.

- The important thing to realize here is that there is no combination of old segments that can cause the protocol to fail and have a connection set up by accident when no one wants it.

3.CONNECTION RELEASE:

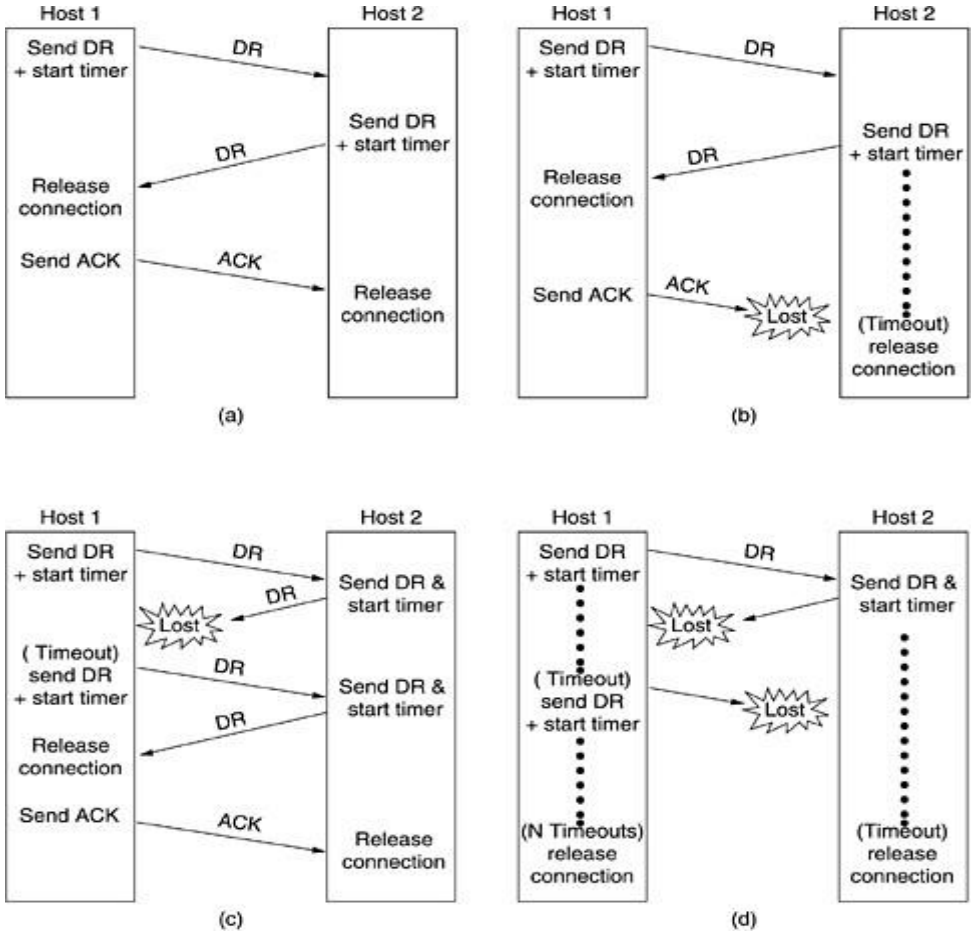
A connection is released using either asymmetric or symmetric variant. But, the improved protocol for releasing a connection is a 3-way handshake protocol.

There are two styles of terminating a connection:

1) Asymmetric release and
2) Symmetric release.

Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken. **Symmetric release** treats the connection as two separate unidirectional connections and requires each one to be released separately.

Fig-(a)	Fig-(b)	Fig-(c)	Fig-(d)
One of the user sends a DISCONNECTION REQUEST TPDU in order to initiate connection release. When it arrives, the recipient sends back a DR-TPDU, too, and starts a timer. When this DR arrives, the original sender sends back an ACK-TPDU and releases the connection. Finally, when the ACK-TPDU arrives, the receiver also releases the connection.	Initial process is done in the same way as in fig-(a). If the final ACK-TPDU is lost, the situation is saved by the timer. When the timer is expired, the connection is released.	If the second DR is lost, the user initiating the disconnection will not receive the expected response, and will timeout and starts all over again.	Same as in fig-(c) except that all repeated attempts to retransmit the DR is assumed to be failed due to lost TPDUs. After 'N' entries, the sender just gives up and releases the connection.



4.FLOW CONTROL AND BUFFERING:

Flow control is done by having a sliding window on each connection to keep a fast transmitter from over running a slow receiver. Buffering must be done by the sender, if the network service is unreliable. The sender buffers all the TPDUs sent to the receiver. The buffer size varies for different TPDUs.

They are:

a) Chained Fixed-size Buffers
b) Chained Variable-size Buffers
c) One large Circular Buffer per Connection

(a). Chained Fixed-size Buffers:

If most TPDUs are nearly the same size, the buffers are organized as a pool of identical size buffers, with one TPDU per buffer.

(b). Chained Variable-size Buffers:

This is an approach to the buffer-size problem. i.e., if there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, some problems may occur:

- If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
- If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs.

To overcome these problems, we employ variable-size buffers.

(c). One large Circular Buffer per Connection:

A single large circular buffer per connection is dedicated when all connections are heavily loaded.

1. Source Buffering is used for low band width bursty traffic
2. Destination Buffering is used for high band width smooth traffic.
3. Dynamic Buffering is used if the traffic pattern changes randomly.

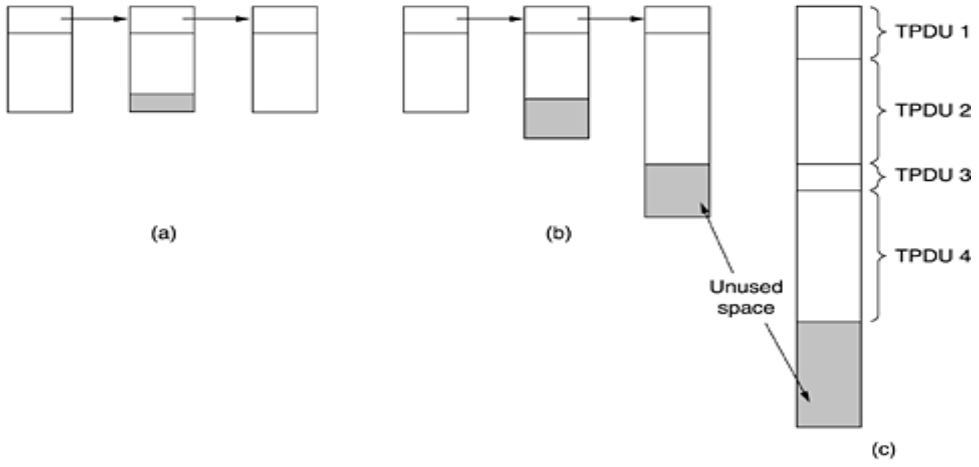


Figure 4.7. (a) Chained fixed-size buffers. (b) Chained variable-sized buffers. (c) One large circular buffer per connection.

5.MULTIPLEXING:

In networks that use virtual circuits within the subnet, each open connection consumes some table space in the routers for the entire duration of the connection. If buffers are dedicated to the virtual circuit in each router as well, a user who left a terminal logged into a remote machine, there is need for multiplexing. There are 2 kinds of multiplexing:

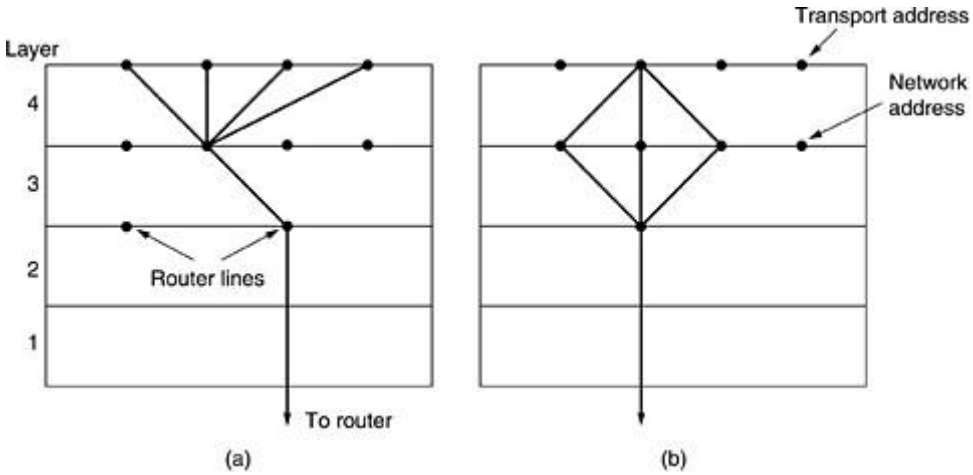


Figure 4.8. (a) Upward multiplexing. (b) Downward multiplexing

(a). <u>UP-WARD MULTIPLEXING:</u>

In the below figure, all the 4 distinct transport connections use the same network connection to the remote host. When connect time forms the major component of the carrier's bill, it is up to the transport layer to group port connections according to their destination and map each group onto the minimum number of port connections.

(b). <u>DOWN-WARD MULTIPLEXING:</u>

- If too many transport connections are mapped onto the one network connection, the performance will be poor.
- If too few transport connections are mapped onto one network connection, the service will be expensive.

The possible solution is to have the transport layer open multiple connections and distribute the traffic among them on round-robin basis, as indicated in the below figure:
With 'k' network connections open, the effective band width is increased by a factor of 'k'.

<u>TRANSPORT</u> PROTOCOLS - UDP

The Internet has two main protocols in the transport layer, **a connectionless protocol** and a **connection-oriented** one. The protocols complement each other. The connectionless protocol is **UDP.** It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

The connection-oriented protocol is **TCP.** It does almost everything. It makes connections and adds reliability with retransmissions, along with flow control and congestion control, all on behalf of the

applications that use it. Since UDP is a transport layer protocol that typically runs in the operating system and protocols that use UDP typically run in user s pace, these uses might be considered applications.

INTROUCTION TO UDP

- The Internet protocol suite supports a connectionless transport protocol called UDP (User Datagram Protocol). UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- UDP transmits segments consisting of an 8-byte header followed by the pay-load. The two ports serve to identify the end-points within the source and destination machines.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when the BIND primitive. Without the port fields, the transport layer would not know what to do with each incoming packet. With them, it delivers the embedded segment to the correct application.

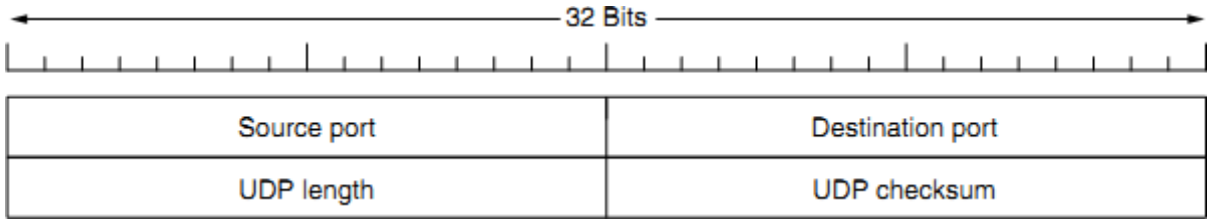


Fig 4.9: The UDP header

Source port, destination port: Identifies the end points within the source and destination machines.
UDP length: Includes 8-byte header and the data
UDP checksum: Includes the UDP header, the UDP data padded out to an even number of bytes if need be. It is an optional field

REMOTE PROCEDURE CALL

- In a certain sense, sending a message to a remote host and getting a reply back is like making a function call in a programming language. This is to arrange request-reply interactions on networks to be cast in the form of procedure calls.
- For example, just imagine a procedure named *get IP address* (*host name*) that works by sending a UDP packet to a DNS server and waiting or the reply, timing out and trying again if one is not forthcoming quickly enough. In this way, all the details of networking can be hidden from the programmer.

- RPC is used to call remote programs using the procedural call. When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on 2.

- Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer. This technique is known as **RPC** (**Remote Procedure Call**) and has become the basis for many networking applications.

Traditionally, the calling procedure is known as the **client** and the called procedure is known as the **server.**

- In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub**,that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local.

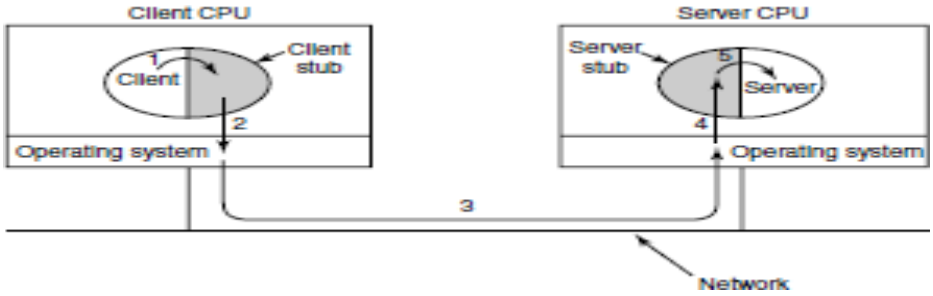


Fig 4.10: Steps in making a RPC

Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.
Step 3 is the operating system sending the message from the client machine to the server machine.
Step 4 is the operating system passing the incoming packet to the server stub.
Step 5 is the server stub calling the server procedure with the **unmarshaled** parameters. The reply traces the same path in the other direction.

The key item to note here is that the client procedure, written by the user, just makes a normal (i.e., local) procedure call to the client stub, which has the same name as the server procedure. Since the client procedure and client stub are in the same address space, the parameters are passed in the usual way.

Similarly, the server procedure is called by a procedure in its address space with the parameters it expects. To the server procedure, nothing is unusual. In this way, instead of I/O being done on sockets, network communication is done by faking a normal procedure call. With RPC, passing pointers is impossible because the client and server are in different address spaces.

TCP (TRANSMISSION CONTROL PROTOCOL)

It was specifically designed to provide a reliable end-to end byte stream over an unreliable network. It was designed to adapt dynamically to properties of the inter network and to be robust in the face of many kinds of failures.

Each machine supporting TCP has a TCP transport entity, which accepts user data streams from local processes, breaks them up into pieces not exceeding 64kbytes and sends each piece as a separate IP datagram. When these datagrams arrive at a machine, they are given to TCP entity, which reconstructs the original byte streams. It is up to TCP to time out and retransmits them as needed, also to reassemble datagrams into messages in proper sequence.

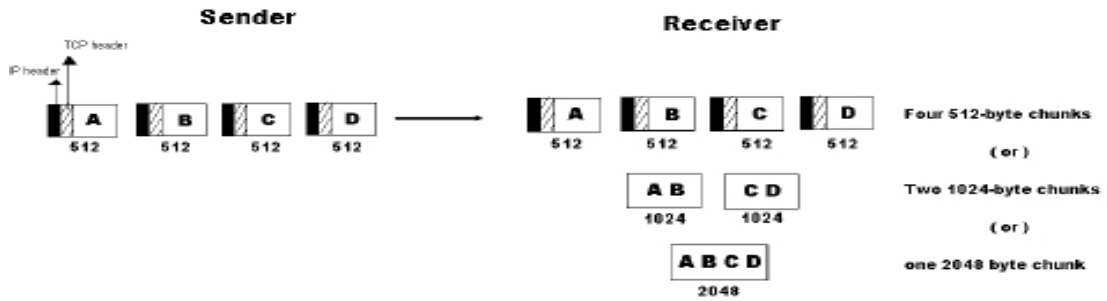
The different issues to be considered are:

1. The TCP Service Model
2. The TCP Protocol
3. The TCP Segment Header
4. The Connection Management
5. TCP Transmission Policy
6. TCP Congestion Control
7. TCP Timer Management.

The TCP Service Model

- TCP service is obtained by having both the sender and receiver create end points called **SOCKETS**
- Each socket has a socket number(address)consisting of the IP address of the host, called a **"PORT"** (= TSAP)
- To obtain TCP service a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine
- All TCP connections are full duplex and point to point i.e., multicasting or broadcasting is not supported.
- A TCP connection is a byte stream, not a message stream i.e., the data is delivered as chunks

E.g.: 4 * 512 bytes of data is to be transmitted.



Sockets:

A socket may be used for multiple connections at the same time. In other words, 2 or more connections may terminate at same socket. Connections are identified by socket identifiers at same socket. Connections are identified by socket identifiers at both ends. Some of the sockets are listed below:

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Ports: Port numbers below 256 are called Well- known ports and are reserved for standard services.
***Eg*:**

PORT-21	To establish a connection to a host to transfer a file using FTP
PORT-23	To establish a remote login session using TELNET

The TCP Protocol

- A key feature of TCP, and one which dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number.
- When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host blasting away at full speed took over 1 week to cycle through the sequence numbers.
- The basic protocol used by TCP entities is the **sliding window protocol**.
- When a sender transmits a segment, it also starts a timer.
- When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, otherwise without data) bearing an acknowledgement number equal to the next sequence number it expects to receive.
- If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header

Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. After the options, if any, up to 65,535 - 20 - 20 = 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header. Segments without any data are legal and are commonly used for acknowledgements and control messages.

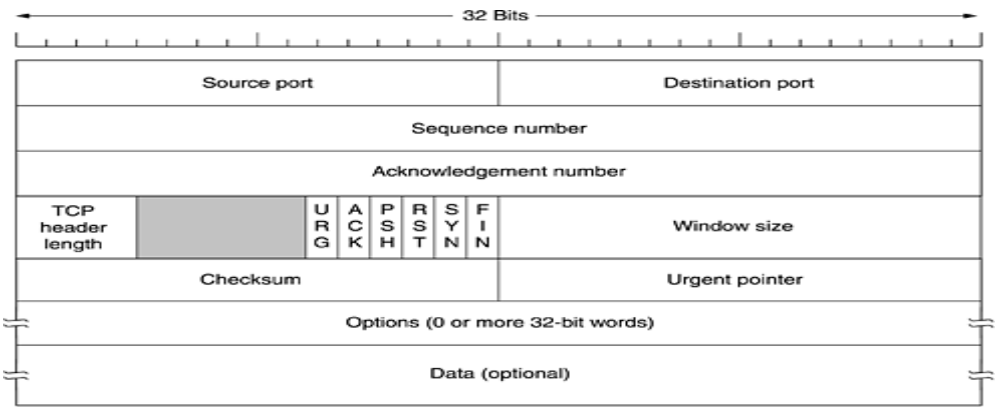


Fig 4.11: The TCP Header

Source Port, Destination Port : Identify local end points of the connections

Sequence number: Specifies the sequence number of the segment

Acknowledgement Number: Specifies the next byte expected.

TCP header length: Tells how many 32-bit words are contained in TCP header

URG: It is set to 1 if URGENT pointer is in use, which indicates start of urgent data.

ACK: It is set to 1 to indicate that the acknowledgement number is valid.

PSH: Indicates pushed data

RST: It is used to reset a connection that has become confused due to reject an invalid segment or refuse an attempt to open a connection.

FIN: Used to release a connection.

SYN: Used to establish connections.

TCP Connection Establishment

To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives, either specifying a specific source or nobody in particular.

The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password).

The CONNECT primitive sends a TCP segment with the *SYN* bit on and *ACK* bit off and waits for a response.

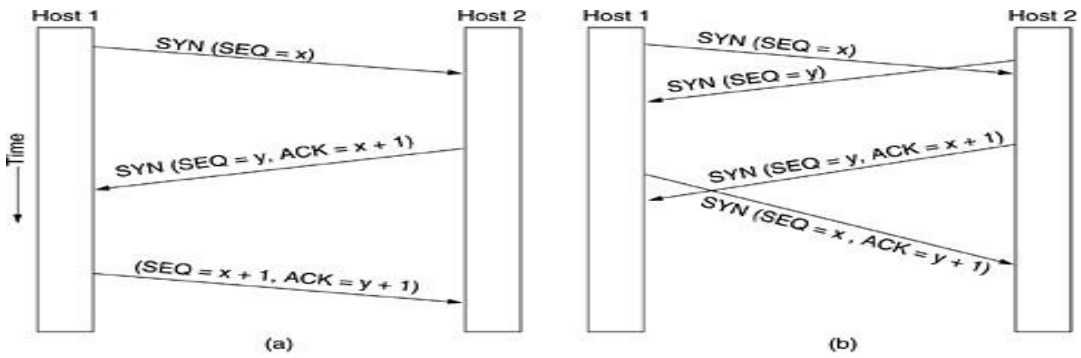


Fig 4.12: a) TCP Connection establishment in the normal case b) Call Collision

TCP Connection Release

- Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections.
- Each simplex connection is released independently of its sibling. To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit.
- When the *FIN* is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the other direction, however.
- When both directions have been shut down, the connection is released.
- Normally, four TCP segments are needed to release a connection, one *FIN* and one *ACK* for each direction. However, it is possible for the first *ACK* and the second *FIN* to be contained in the same segment, reducing the total count to three.

TCP Connection Management Modeling

The steps required establishing and release connections can be represented in a finite state machine with the 11 states listed in Fig. 4.13. In each state, certain events are legal. When a legal event happens, some action may be taken. If some other event happens, an error is reported.

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Figure 4.13. The states used in the TCP connection management finite state machine.

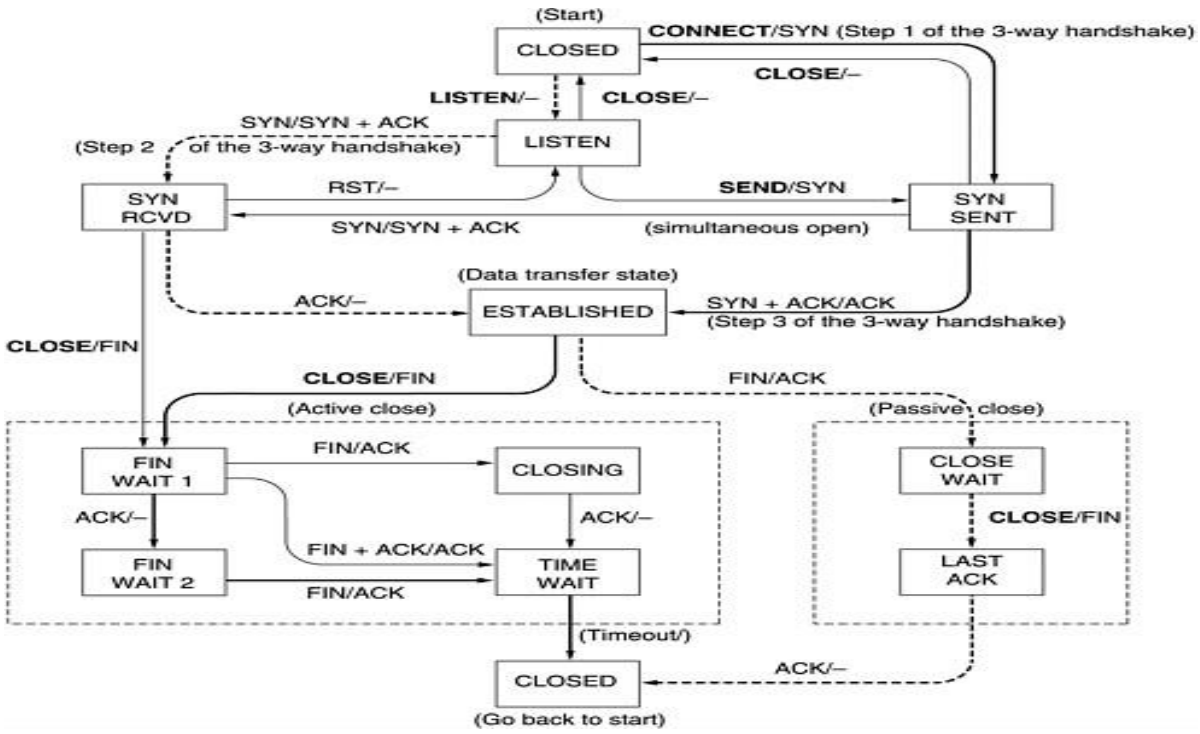


Figure 4.14 - TCP connection management finite state machine.

TCP Connection management from server's point of view:

1. The server does a **LISTEN** and settles down to see who turns up.
2. When a **SYN** comes in, the server acknowledges it and goes to the **SYNRCVD** state
3. When the servers **SYN** is itself acknowledged the 3-way handshake is complete and server goes to the **ESTABLISHED** state. Data transfer can now occur.
4. When the client has had enough, it does a close, which causes a **FIN** to arrive at the server [dashed box marked passive close].
5. The server is then signaled.
6. When it too, does a **CLOSE**, a **FIN** is sent to the client.
7. When the client's acknowledgement shows up, the server releases the connection and deletes the connection record.

TCP CONGESTION CONTROL:

TCP does to try to prevent the congestion from occurring in the first place in the following way:

When a connection is established, a suitable window size is chosen and the receiver specifies a window based on its buffer size. If the sender sticks to this window size, problems will not occur due to buffer overflow at the receiving end. But they may still occur due to internal congestion within the network. Let's see this problem occurs.

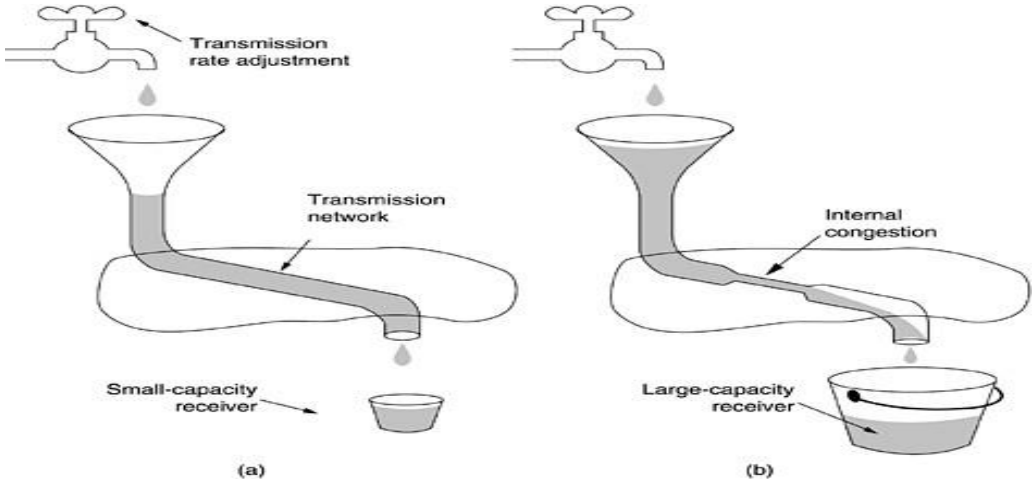


Figure 4.16. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

In fig (a): We see a thick pipe leading to a small- capacity receiver. As long as the sender does not send more water than the bucket can contain, no water will be lost.

In fig (b): The limiting factor is not the bucket capacity, but the internal carrying capacity of the n/w. if too much water comes in too fast, it will backup and some will be lost.

- When a connection is established, the sender initializes the congestion window to the size of the max segment in use our connection.
- It then sends one max segment .if this max segment is acknowledged before the timer goes off, it adds one segment s worth of bytes to the congestion window to make it two maximum size segments and sends 2 segments.
- As each of these segments is acknowledged, the congestion window is increased by one max segment size.
- When the congestion window is 'n' segments, if all 'n' are acknowledged on time, the congestion window is increased by the byte count corresponding to 'n' segments.
- The congestion window keeps growing exponentially until either a time out occurs or the receiver's window is reached.
- The internet congestion control algorithm uses a third parameter, the **"threshold"** in addition to receiver and congestion windows.

Different congestion control algorithms used by TCP are:

- RTT variance Estimation.
- Exponential RTO back-off — Re-transmission Timer Management
- Karn's Algorithm
- Slow Start
- Dynamic window sizing on congestion
- Fast Retransmit — Window Management
- Fast Recovery

MODULE V: APPLICATION LAYER

Principles of Network Applications-Meaning:

- Write programs that: run on (different) end systems communicate over network
- e.g. web server software communicates with browser software
- no need to write software for network-core devices
- network-core devices do not run user applications
- Examples of network-applications:
  1. Web
  2. File transfers
  3. E-mail
  4. P2Pfile sharing
  5. Social networking(Facebook,Twitter)
  6. Video distribution(YouTube)
  7. Real-time video conferencing(Skype)
  8. On-line games (World of War craft)
- For ex:In the Web application, there are 2different programs:
  1) The browser program running in the user's host (Laptop or Smartphone).
  2) The Web-server program running in the Web-server host.

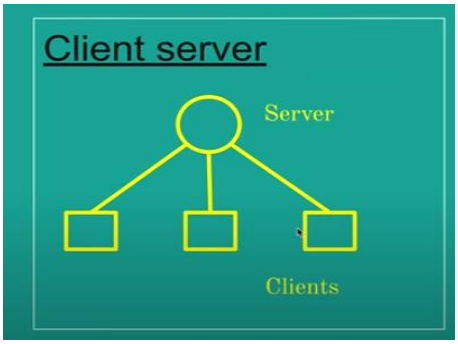
Principles of Network Applications are:

- Network Application Architectures
- Process Communication
- Transport Service Available to Application
- Transport Service provided by Internet
- Application Layer Protocol

Network Application Architectures

- Two approaches for developing an application:
  1) Client-Server architecture 2) P2P(Peer to Peer) architecture

Client-Server Architecture:



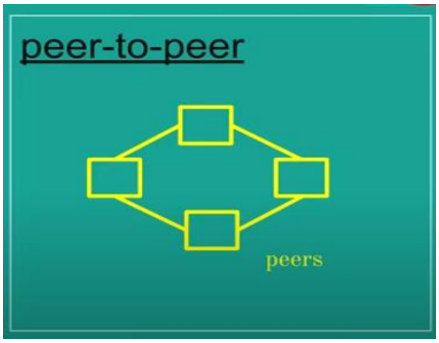
- **Server:** The server is always-on host, while a client can be randomly run.
  The server has permanent IP address.
  Often in data centers, for scaling
- **Clients:** contact, communicate with server.
  May have dynamic IP addresses.
  Do not communicate directly with each other.

- The server is listening on the network and a client initializes the communication.
- Upon the requests from a client, the server provides certain services to the client.
- Usually, there is no communication between two clients.
- A client contacts the server by sending a packet to the server's IP address.
- A server is able to communicate with many clients.
- The applications such as FTP, telnet, Web, e-mail uses client-server architecture.

Data Center

- Earlier, client-server architecture had a single-server host.
- But now, a single-server host is unable to keep up with all the requests from large no. of clients.
- For this reason, data-center is used.
- A data-center contains a large number of hosts.
- A data-center is used to create a powerful virtual server.
- In datecenter, hundreds of servers must be powered and maintained.
- For example:
  - Google has around 50 data-centers distributed around the world.
  - These 50data-centers handle search, YouTube, Gmail, and other services.

P2PArchitecture



- There is no dedicated server.
- Pairs of hosts are called peers.
- The peers communicate directly with each other.
- The peers are not owned by the service-provider.Rather, the peers are laptops controlled by users. Examples include file sharing (Bit Torrent), Internet telephone (Skype) etc.
- Main feature of P2P architectures: self-scalability.
- For ex:In a P2P file-sharing system,
  - Each peer generates workload by requesting files.
  - Each peer also adds service-capacity to the system by distributing files too their peers.
- Advantage: Cost effective‘.’Normally, server-infrastructure &server bandwidth are not required.
- Three challenges of the P2Papplications:

1) ISP Friendly

- Most residential ISPs have been designed for a symmetrical bandwidth usage.
- A symmetrical bandwidth means there is more downstream-traffic than upstream-traffic.
- But P2P applications shift upstream-traffic from servers to residential ISPs, which stress on the ISPs.

2) Security

- Since the highly distribution and openness,P2P applications can be a challenge to security.

3) Incentive

- Success of P2P depends on convincing users to volunteer bandwidth & resources to the applications.

Processes Communicating

Process

- A process is an instance of a program running in a computer.
- The processes may run on the 1)same system or 2)different systems.

1) The processes running on the same end-system can communicate with each other using IPC(Inter Process Communication).
2) The processes running on the different end-systems can communicate by exchanging messages.

i) A sending-process creates and sends messages into the network.
ii) A receiving-process receives the messages and responds by sending messages back.

Client & Server Processes

- A network-application consists of pairs of processes:

1) The process that initiates the communication is labeled as the client.
2) The process that waits to be contacted to begin the session is labeled as the server.

- For example:

1) In Web application, a client-browser process communicates with a Web-server-process.
2) In P2P file system, a file is transferred from a process in one peer to a process in another peer.

Interface between the Process and the Computer Network Socket

- Any message sent from one process to another must go through the underlying-network.
- A process sends/receives message through a software-interface of underlying-network called socket.
- Socket is an API between the application-layer and the transport layer within a host.
- The application-developer has complete control at the application-layer side of the socket.
- But, the application-developer has little control of the transport-layer side of the socket. For ex: The application-developer can control:

1) The choice of transport-protocol: TCP or UDP. (API Application Programming Interface)
2) The ability to fix parameters such as maximum-buffer & maximum-segment-sizes.

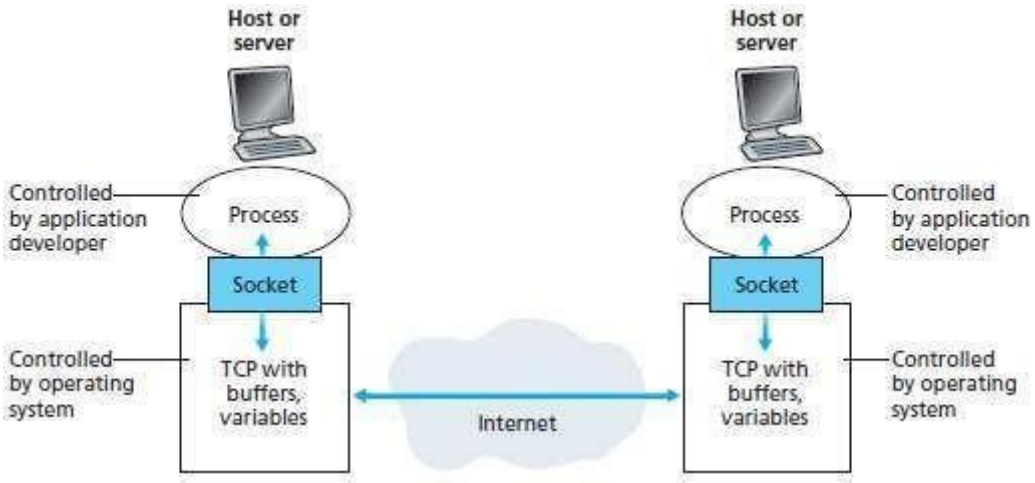


Figure1.2: Application processes, sockets, and transport-protocol

Addressing Processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - *A:* no, *many* processes can be running on same host

- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80

Transport Services Available to Applications

1. Reliable Data Transfer
2. Throughput
3. Timing
4. Security

Reliable Data Transfer:

- Reliable means guaranteeing the data from the sender to the receiver is delivered correctly. For ex: TCP provides reliable service to an application.
- Unreliable means the data from the sender to the receiver may never arrive. For ex: UDP provides unreliable service to an application.
- Unreliability may be acceptable for loss-tolerant applications, such as multimedia applications.
- In multimedia applications, the lost data might result in a small glitch in the audio/video.

Throughput:

- Throughput is the rate at which the sending-process can deliver bits to the receiving-process.
- Since other hosts are using the network, the throughput can fluctuate with time.
- Two types of applications:

1) Bandwidth Sensitive Applications

These applications need a guaranteed throughput. For ex: Multimedia applications.
Some transport-protocol provides guaranteed throughput at some specified rate(r bits/sec).

2) Elastic Applications

These applications may not need a guaranteed throughput. For ex:Electronic mail, File transfer & Web transfers.

Timing:

- A transport-layer protocol can provide timing-guarantees.
- For ex: guaranteeing every bit arrives at the receiver in less than 100msec.
- Timing constraints are useful for real-time applications such as
  →Internet telephony
  →Virtual environments
  →Tele conferencing and
  →Multi player games

Security:

- A transport-protocol can provide one or more security services.
- For example,

1) In the sending host, a transport-protocol can encrypt all the transmitted-data.
2) In the receiving host, the transport-protocol can decrypt the received-data.

Transport Services Provided by the Internet

- The Internet makes two transport-protocols available to applications, UDP and TCP.
- An application-developer who creates a new network-application must use either: UDP or TCP.
- Both UDP&TCP offers a different set of services to the invoking applications.
- Table1.1shows the service requirements for some selected applications.

Application	**Data Loss**	**Throughput Time**	**Sensitive**
File transfer /download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic(few kbps)	No
Internet-telephony/ Video-conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video:10 kbps–5 Mbps	Yes:100s of ms
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10kbps	Yes:100sof ms
Instant messaging	No loss	Elastic	Yes and no

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *connection-oriented:* setup required between client and server processes
- *does not provide:* timing, minimum throughput guarantee, security

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup.

Q: why bother? *Why* is there a UDP?

Application Layer Protocol

- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

The Web and HTTP:
Overview of HTTP Web

- A web-page consists of objects (HTML Hyper Text Markup Language).
- An object is a file such as an HTML file, a JPEG image, a Java applet, a video clip.
- The object is addressable by a single URL(URL Uniform Resource Locator).
- Most Web-pages consist of a base HTML file & several referenced objects.
- For example:

If a Web-page contains HTML text and five JPEG images; then the Web-page has six objects:

1) Base HTML file and
2) Five images.

- The base HTML file references the other objects in the page with the object's URLs.
- URL has 2 components:

1) The host name of the server that houses the object and
2) The object's path name.

- For example: "http://www.someSchool.edu/some Department/picture.gif"
- In above URL,

1) Hostname="www.someSchool.edu"
2) Pathname="/some Department /picture.gif".

HTTP

- HTTP is Web's application-layer protocol (Figure1.3) (HTTP Hyper Text Transfer Protocol).
- HTTP defines
  →how clients request Web-pages from servers and
  →how servers transfer Web-pages to clients.

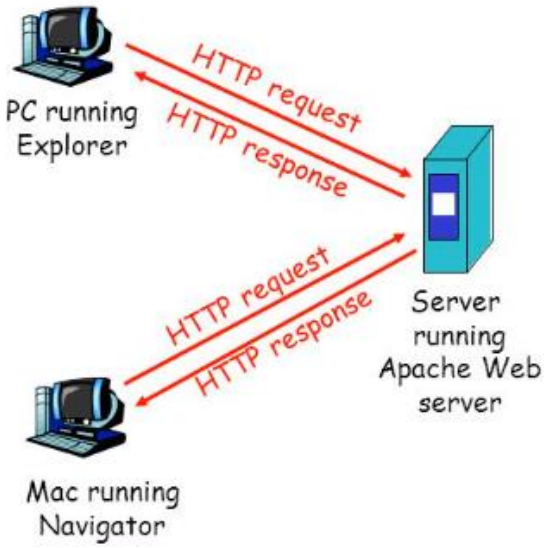


Figure1.3: HTTP request-response behavior

- When a user requests a Web-page, the browser sends HTTP request to the server.
- Then, the server responds with HTTP response that contains the requested-objects.
- HTTP uses TCP as its underlying transport-protocol.
- The HTTP client first initiates a TCP connection with the server.
- After connection setup, the browser and the server-processes access TCP through their sockets.
- HTTP is a stateless protocol.
- Stateless means the server sends requested-object to client w/o storing state-info about the client.
- HTTP uses the client-server architecture:

1) Client

- Browser that requests receive and displays Web objects.

2) Server

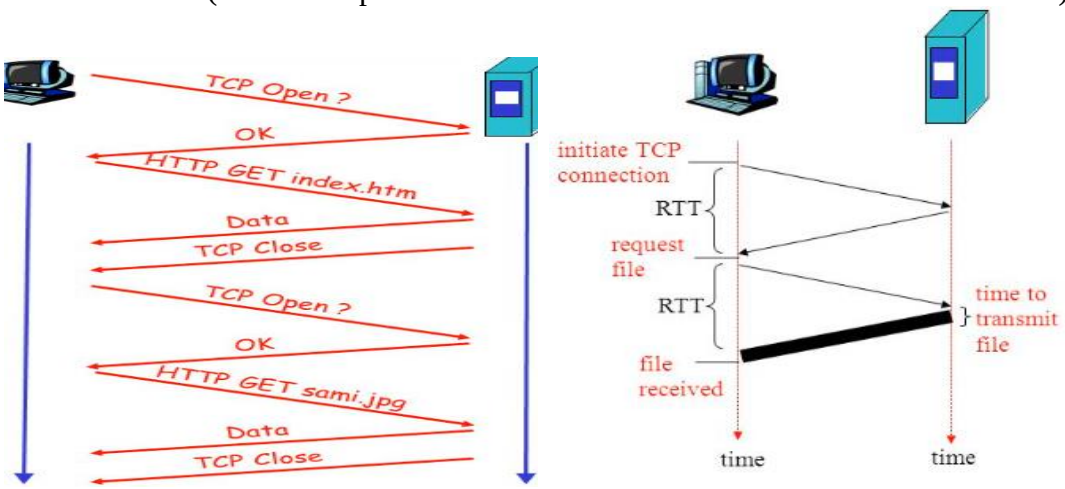
- Web-server sends objects in response to requests.

HTTP Connections:

Non-Persistent & Persistent Connections

Non-Persistent

- In non-persistent connection HTTP, there can be at most one object that can be sent over a single TCP connection. This means that for each object that is to be sent from source to destination, a new Connection will be created. HTTP/1.0 is the version of HTTP that uses a non-persistent connection.
- Non-persistent connection HTTP requires 2 RTT (round trip time) for each object that is to be Transmitted (1 RTT to open the connection and 1 RTT for transmission of data).



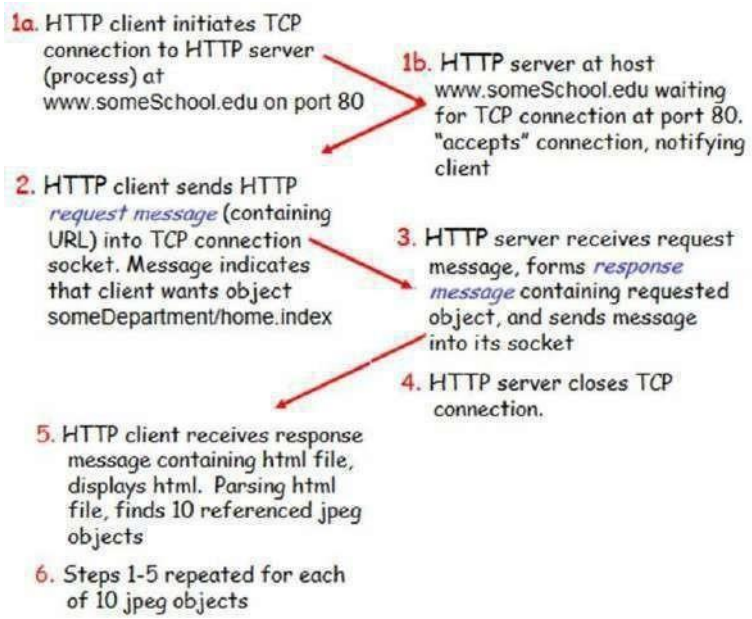
Total transmission time=2RTT+file transmission time

Advantages of Non-Persistent HTTP

- It does not lead to wastage of resources since the connection is opened only when some data needs to be sent over it.
- It is more secure than persistent HTTP since after sending data over the connection, the connection gets terminated and nothing can be transmitted over it once it gets terminated.

Disadvantages of Non-Persistent HTTP

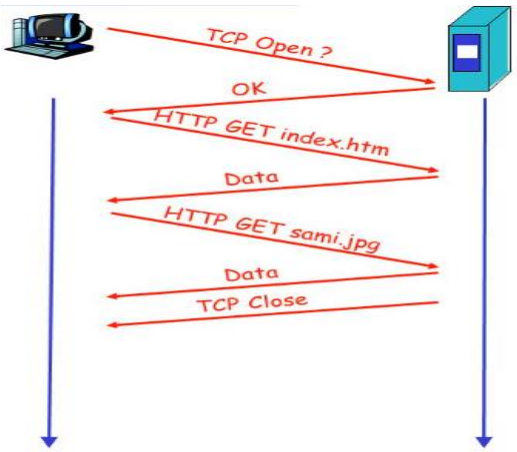
- It needs to maintain an extra overhead to open a TCP connection each time some data needs to be transmitted over it.
- It has a slow start because of the opening of a TCP connection on every data transmission.
- Here is how it works:



HTTP with Persistent Connections

- With persistent connections, the server leaves the TCP connection open after sending responses.
- Hence, subsequent requests & responses b/w same client & server can be sent over same connection
- The server closes the connection only when the connection is not used for a certain amount of time.
- Default mode of HTTP: Persistent connections with pipelining.
- Advantages:

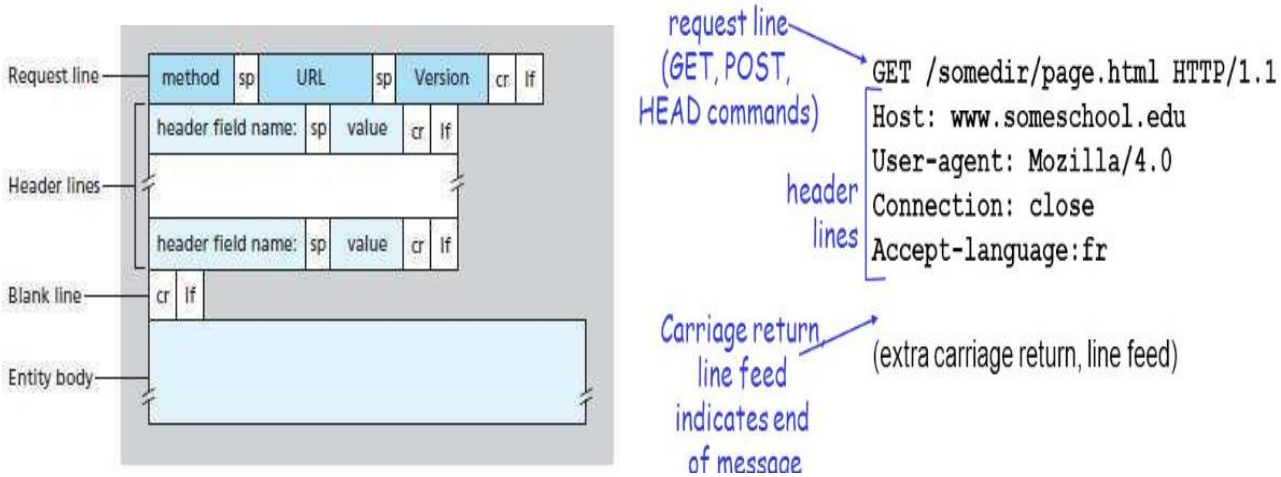
1) This method requires only one RTT for all the referenced-objects.
2) The performance is improved by 20%.



HTTP Message Format

- Two types of HTTP messages: 1) Request-message and 2) Response-message.

HTTP Request Message



General format of an HTTP request-message

- Three sections:
  - ✓ Request-line
  - ✓ Header-line and
  - ✓ Carriage return.
- The first line of message is called the request-line. The subsequent lines are called the header-lines.
- The request-line contains 3 fields. The meaning of the fields is as follows:

1) Method
   - ➢ "GET": This method is used when the browser requests an object from the server.
2) URL
   - ➢ "/somedir/page.html":This is the object requested by the browser.
3) Version
   - ➢ "HTTP/1.1": This is version used by the browser.

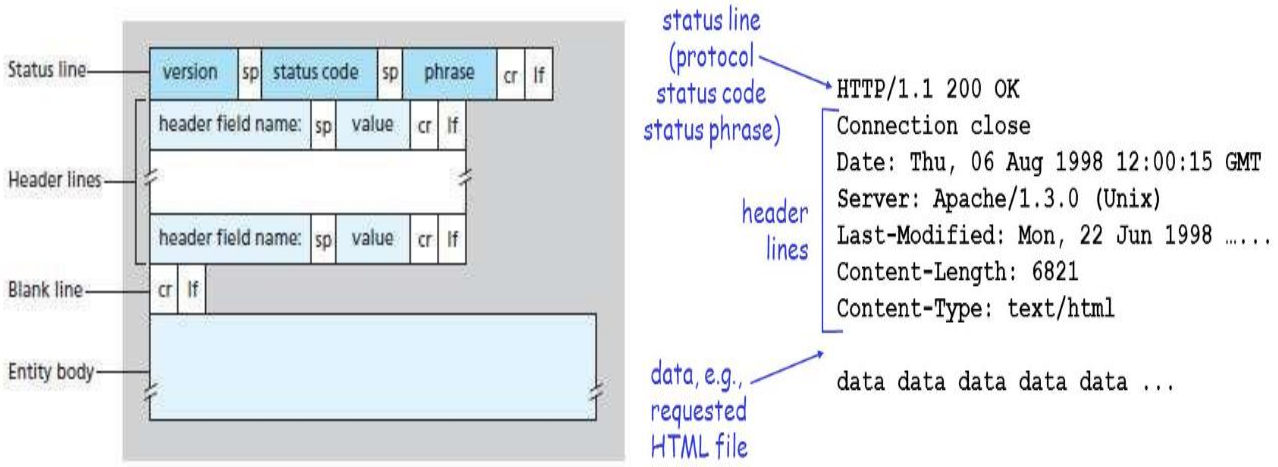
- The request-message contains 4 header-lines. The meaning of the header-lines is as follows:

1) "Host:www.someschool.edu" specifies the host on which the object resides.
2) "Connection: close" means requesting a non-persistent connection.
3) "User-agent: Mozilla/5.0" means the browser used is the Firefox.
4) "Accept-language:fr" means French is the preferred language.

- The method field can take following values: GET, POST, HEAD, PUT and DELETE.

1) GET is used when the browser requests an object from the server.
2) POST is used when the user fills out a form & sends to the server.
3) HEAD is identical to GET except the server must not return a message-body in the response.
4) PUT is used to upload objects to servers.
5) DELETE allows an application to delete an object on a server.

HTTP Response Message



General format of an HTTP response-message

- Three sections:
  1) Status line
  2) Header-lines and
  3) Data (Entity body).

- The status line contains 3 fields:

1) Protocol version
2) Status-code and
3) Status message.

- Some common status-codes and associated messages include:

1) 200 OK: Standard response for successful HTTP requests.
2) 400 Bad Request: The server cannot process the request due to a client error.
3) 404 Not Found: The requested resource cannot be found.

- The meaning of the Status line is as follows:
  "HTTP/1.1 200 OK": This line indicates the server is using HTTP/1.1&that everything is OK.
- The response-message contains 6 header-lines. The meaning of the header-lines is as follows:

1) Connection: This line indicates browser requesting a non-persistent connection.
2) Date: This line indicates the time & date when the response was sent by the server.
3) Server: This line indicates that the message was generated by an Apache Web-server.
4) Last-Modified: This line indicates the time & date when the object was last modified.
5) Content-Length: This line indicates the number of bytes in the sent-object.
6) Content-Type: This line indicates that the object in the entity body is HTML text.

User-Server Interaction: Cookies

- Cookies refer to a small text file created by a Web-site that is stored in the user's computer.
- Cookies are stored either temporarily for that session only or permanently on the hard disk.
- Cookies allow Web-sites to keep track of users.
- Cookie technology has four components:

1) A cookie header-line in the HTTP response-message.
2) A cookie header-line in the HTTP request-message.
3) A cookie file kept on the user 'send-system and managed by the user's browser.
4) A back-end database at the Web-site.

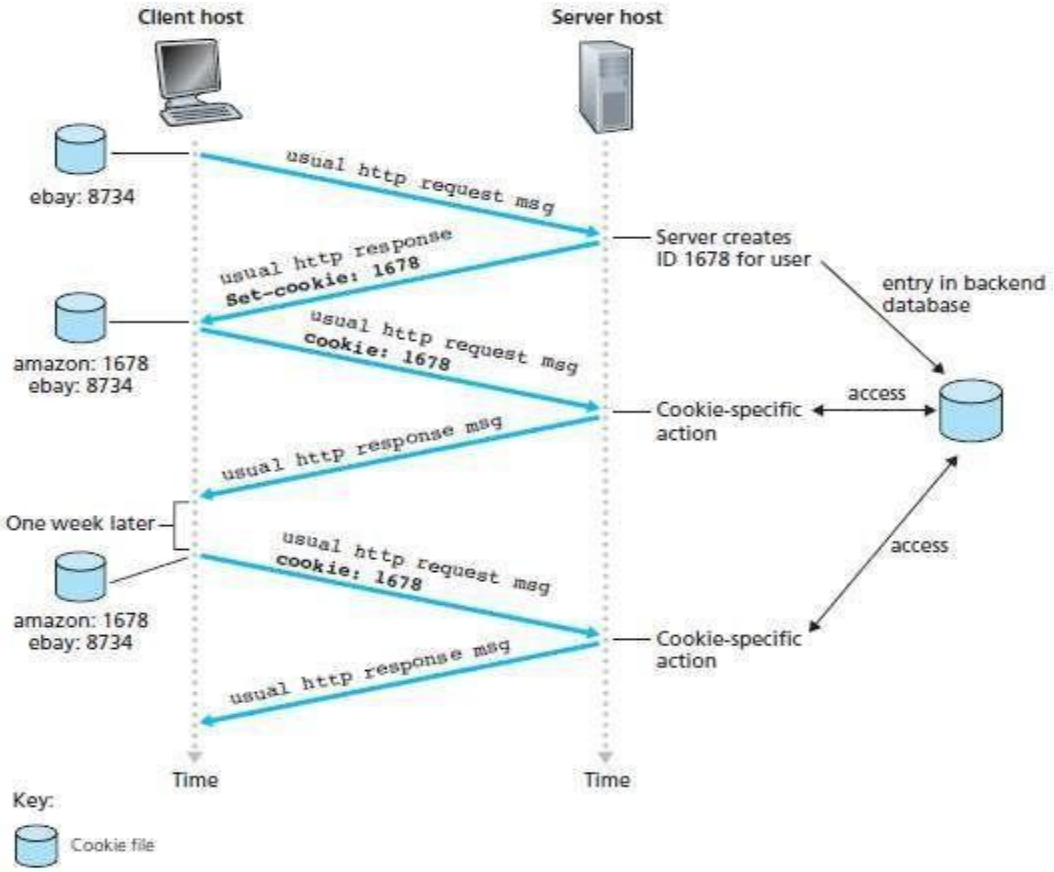


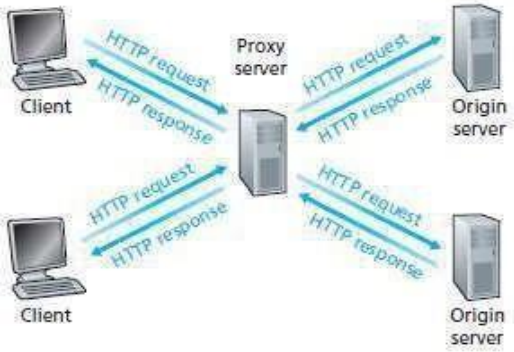
Figure1.7:Keeping user state with cookies

- Here is how it works (Figure1.7):

1) When a user first time visits a site, the server
   →creates a unique identification number (1678) and
   →creates an entry in its back-end database by the identification number.
2) The server then responds to user's browser.
   - HTTP response includes Set-cookie: header which contains the identification number(1678)
3) The browser then stores the identification number in to the cookie-file.
4) Each time the user requests a Web-page, the browser
   →extracts the identification number from the cookie file, and
   →puts the identification number in the HTTP request.
5) In this manner, the server is able to track user's activity at the web-site.

Web Caching

- A Web-cache is a network entity that satisfies HTTP requests on the behalf of an original Web-server.
- The Web-cache has disk-storage.
- The disk-storage contains copies of recently requested-objects.



Here is how it works

1) The user's HTTP requests are first directed to the web-cache.
2) If the cache has the object requested, the cache returns the requested-object to the client.
3) If the cache does not have the requested-object, then the cache
   →connects to the original server and
   →asks for the object.
4) When the cache receives the object, the cache
   →stores a copy of the object in local-storage and
   →sends a copy of the object to the client.

- A cache acts as both a server and a client at the same time.

1) The cache acts as a server when the cache
   →receives requests from a browser and
   →sends responses to the browser.
2) The cache acts as a client when the cache
   →requests to an original server and
   →receives responses from the origin server.

- Advantages of caching:

1) To reduce response-time for client-request.
2) To reduce traffic coming from institution's access-link to the Internet.
3) To reduce Web-traffic in the Internet.

The Conditional GET

- Conditional GET refers a mechanism that allows a cache to verify that the objects are up to date.
- An HTTP request-message is called conditional GET if

1) Request-message uses the GET method and
2) Request-message includes an If-Modified-Since: header-line.

- The following is an example of using conditional GET:

```
GET /fruit/kiwi.fig HTTP1.1
Host:www.exoriguecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

- The response is:

```
HTTP/1.1   304   Not Modified
Date: Sat, 15 Oct 2011 15:39:29
```

File Transfer: FTP

- FTP is used by the local host to transfer files to or from a remote-host over the network.
- FTP uses client-server architecture.
- FTP uses 2 parallel TCP connections:

1) Control Connection

- The control-connection is used for sending control-information b/w local and remote-hosts.
- The control-information includes:
  →user identification
  → Password
  →commands to change directory and
  →commands to put & get files.

2) Data Connection

➢ The data-connection is used to transfer files.

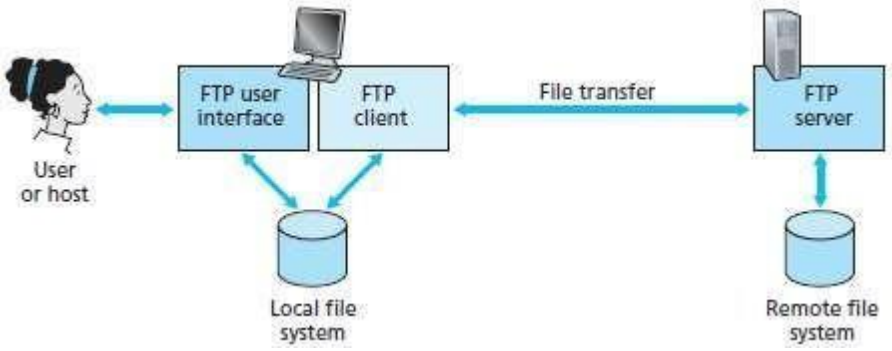


Figure: FTP moves files between local and remote file systems

Figure: Control and data-connections

- Here is how it works:

1) When session starts, the client initiates a control-connection with the server on port 21.
2) The client sends user-identity and password over the control-connection.
3) Then, the server initiates data-connection to the client on port 20.
4) FTP sends exactly one file over the data-connection and then closes the data-connection.
5) Usually, the control-connection remains open throughout the duration of the user-session.
6) But, a new data-connection is created for each file transferred within a session.

- During a session, the server must maintain the state-information about the user.
- For example:
  The server must keep track of the user's current directory.
- Disadvantage:
  Keeping track of state-info limits the no. of sessions maintained simultaneously by a server.

FTP Commands & Replies

Commands:

1) USER username
   ➢ Used to send the user identification to the server.
2) PASS password
   ➢ Used to send the user password to the server.
3) LIST
   ➢ Used to ask the server to send backa list of all the files in the current remote directory.
4) RETR filename
   ➢ Used to retrieve a file from the current directory of the remote-host.
5) STOR filename
   ➢ Used to store a file into the current directory of the remote-host.

Replies:

- Each reply consists of 3-digit numbers followed by optional message.

- For example:

1) 331 Username OK, password required
2) 125 Data-connection already open; transfer starting
3) 425 Can't open data-connection
4) 452 Error writing file

Electronic Mail in the Internet

- E-mail is an asynchronous communication medium in which people send and read messages.
- E-mail is fast, easy to distribute, and inexpensive.
- E-mail has features such as
  →messages with attachments
  → Hyperlinks
  →HTML-formatted text and
  →embedded photos.
- Three major components of any-mail system:

1) User Agents

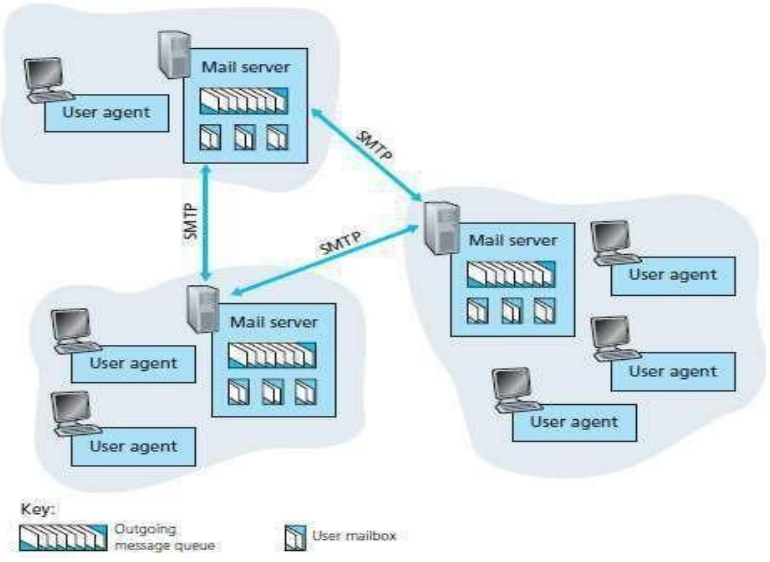
- User-agents allow users to read, reply to, forward, save and compose messages.
- For example : Microsoft Outlook and Apple Mail

2) Mail Servers

- Mail-servers contain mail boxes for users.
- A message is first sent to the sender's mail-server.
- Then, the sender's mail-server sends the message to the receiver's mail-server.
- If the sender's server cannot deliver mail to receiver's server, the sender's server
  →holds the message in a message queue and
  →attempts to transfer the message later.

3) SMTP (Simple Mail Transfer Protocol)

- SMTP is an application-layer protocol used for email.
- SMTP uses TCP to transfer mail from the sender's mail-server to the recipient's mail-server.
- SMTP has two sides:
  1) A client-side, which executes on the sender's mail-server.
  2) A server-side, which executes on the recipient's mail-server.
- Both the client and server-sides of SMTP run on every mail-server.
- When a mail-server receives mail from other mail-servers, the mail-server acts as a server. When a mail-server sends mail to other mail-servers, the mail-server acts as a client.



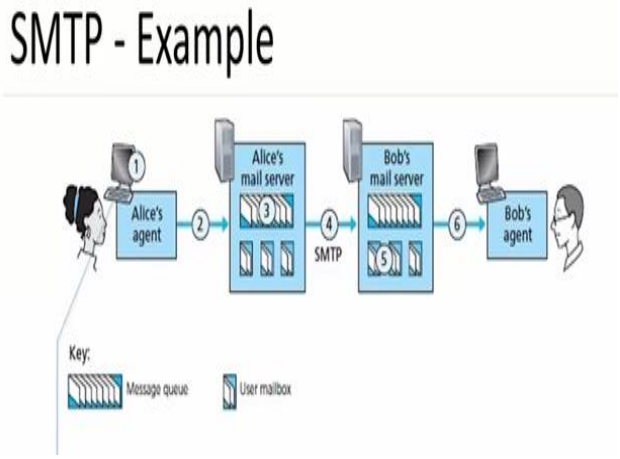
SMTP

- SMTP is the most important protocol of the email system.
- Three characteristics of SMTP(that differs from other applications):

1) Message body uses 7-bit ASCII code only.
2) Normally, no intermediate mail-servers used for sending mail.
3) Mail transmissions across multiple networks through mail relaying.

- Here is how it works:

1) Usually, mail-servers are listening at port 25.
2) The sending server initiates a TCP connection to the receiving mail-server.
3) If the receiver's server is down, the sending server will try later.
4) If connection is established, the client & the server perform application-layer handshaking.
5) Then, the client indicates thee-mail address of the sender and the recipient.
6) Finally, the client sends the message to the server over the same TCP connection.



1. Alice uses user agent to compose message to bob@someschool.edu
2. Alice's user agent sends message to her mail server; message placed in message queue.
3. Client side of SMTP opens TCP connection with Bob's mail server.
4. SMTP client sends Alice's message over the TCP connection.
5. Bob's mail server places the message in Bob's mailbox.
6. Bob invokes his user agent to read message.

Comparison of SMTP with HTTP

1) HTTP is mainly a pull protocol. This is because
   →someone loads information on a web-server and
   →users use HTTP to pull the information from the server.
   - On the other hand, SMTP is primarily a push protocol. This is because
     →the sending mail-server pushes the file to receiving mail-server.
2) SMTP requires each message to be in seven-bit ASCII format.
   - If message contains binary-data, the message has to been coded into 7-bit ASCII format.
   - HTTP does not have this restriction.
3) HTTP encapsulates each object of message in its own response-message.
   - SMTP places all of the message's objects into one message.

Mail Access Protocols:

- Three mail access protocols:

1) Post Office Protocol (POP)
2) Internet Mail Access Protocol (IMAP) and
3) HTTP.

POP

- POP is an extremely simple mail access protocol.
- POP server will listen at port 110.
- Here is how it works:

➢ The user-agent at client's computer opens a TCP connection to the main server.
➢ POP then progresses through three phases:

1) Authentication

➢ The user-agent sends a user name and password to authenticate the user.

2) Transaction

➢ The user-agent retrieves messages.
➢ Also ,the user-agent can
→mark messages for deletion
→remove deletion marks &
→obtain mail statistics.
➢ The user-agent issues commands, and the server responds to each command with a reply.
➢ There are two responses:
i)+OK: used by the server to indicate that the previous command was fine.
ii)–ERR: used by the server to indicate that something is wrong.

3) Update

➢ After user issues a quit command, the mail-server removes all messages marked for deletion.

- Disadvantage:
  The user cannot manage the mails at remote mail-server. For ex: user cannot delete messages.

IMAP

- IMAP is another mail access protocol, which has more features than POP.
- An IMAP server will associate each message with a folder.
- When a message first arrives at server, the message is associated with recipient's INBOX folder
- Then, the recipient can
  →move the message into a new, user-created folder
  →read the message
  →delete the message and
  →search remote folders for messages matching specific criteria.
- An IMAP server maintains user state-information across IMAP sessions.
- IMAP permits a user-agent to obtain components of messages.
  For example, a user-agent can obtain just the message header of a message.

Web-Based E-Mail (HTTP)

- HTTPs are now used for Web-based email accessing.
- The user-agent is an ordinary Web browser.
- The user communicates with its remote-server via HTTP.
- Now, Web-based emails are provided by many companies including Google, Yahoo etc.

DNS—The Internet's Directory Service

- DNS is an internet service that translates domain-names into IP addresses.
  For ex: the domain-name "www.google.com" might translate to IP address "198.105.232.4".
- Because domain-names are alphabetic, they are easier to remember for human being.

- But, the Internet is really based on IP addresses (DNS □ Domain Name System).

Services Provided by DNS:

- **Address Translation:** DNS is an internet service that translates domain-names into IP addresses**.**
- **Host aliasing:** A host with a complicated hostname can have one or more alias names. For example, a hostname such as **relay1.west-coast.enterprise.com** could have, say, two aliases such as **enterprise.com** and **www.enterprise.com**. In this case, the host name relay1.west-coast.enterprise.com is said to be **canonical hostname**. Alias hostnames, when present, are typically more mnemonic than a canonical hostname. DNS can be invoked by an application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
- **Mail server aliasing:** For obvious reasons, it is highly desirable that email addresses be mnemonic. For example, if Bob has an account with Hotmail, Bob's email address might be as simple as **bob@hotmail.com**. However, the hostname of the Hotmail mail server is more complicated and much less mnemonic than simply hotmail.com (e.g., the canonical hostname might be something like **relay1.west-coast.hotmail.com**). DNS can be invoked by a mail application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
- **Load Distribution:** Increasingly, DNS is also being used to perform load distribution among **replicated servers**, such as replicated Web servers. Busy sites, such as cnn.com, are replicated over multiple servers, with each server running on a different end system, and having a different IP address. For replicated Web servers, a set of IP addresses is thus associated with one canonical hostname. The DNS database contains this set of IP addresses. When clients make a DNS query for a name mapped to a set of addresses, the server responds with the entire set of IP addresses.

Overview of How DNS Works

- Distributed database design is more preferred over centralized design because:

1) A Single Point of Failure

- If the DNS server crashes then the entire Internet will stop.

2) Traffic Volume

- A Single DNS Server cannot handle the huge global DNS traffic.
- But with distributed system, the traffic is distributed and reduces overload on server.

3) Distant Centralized Database

- A single DNS server cannot be“close to” all the querying clients.
- If we put the single DNS server in Mysore,
  Then all queries from USA must travel to the other side of the globe.
- This can lead to significant delays.

4) Maintenance

- The single DNS server would have to keep records for all Internet hosts.
- This centralized database has to be updated frequently to account for every new host.

A Distributed, Hierarchical Database

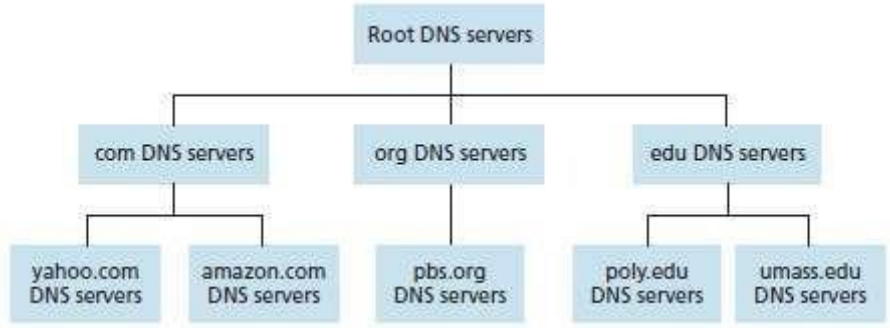


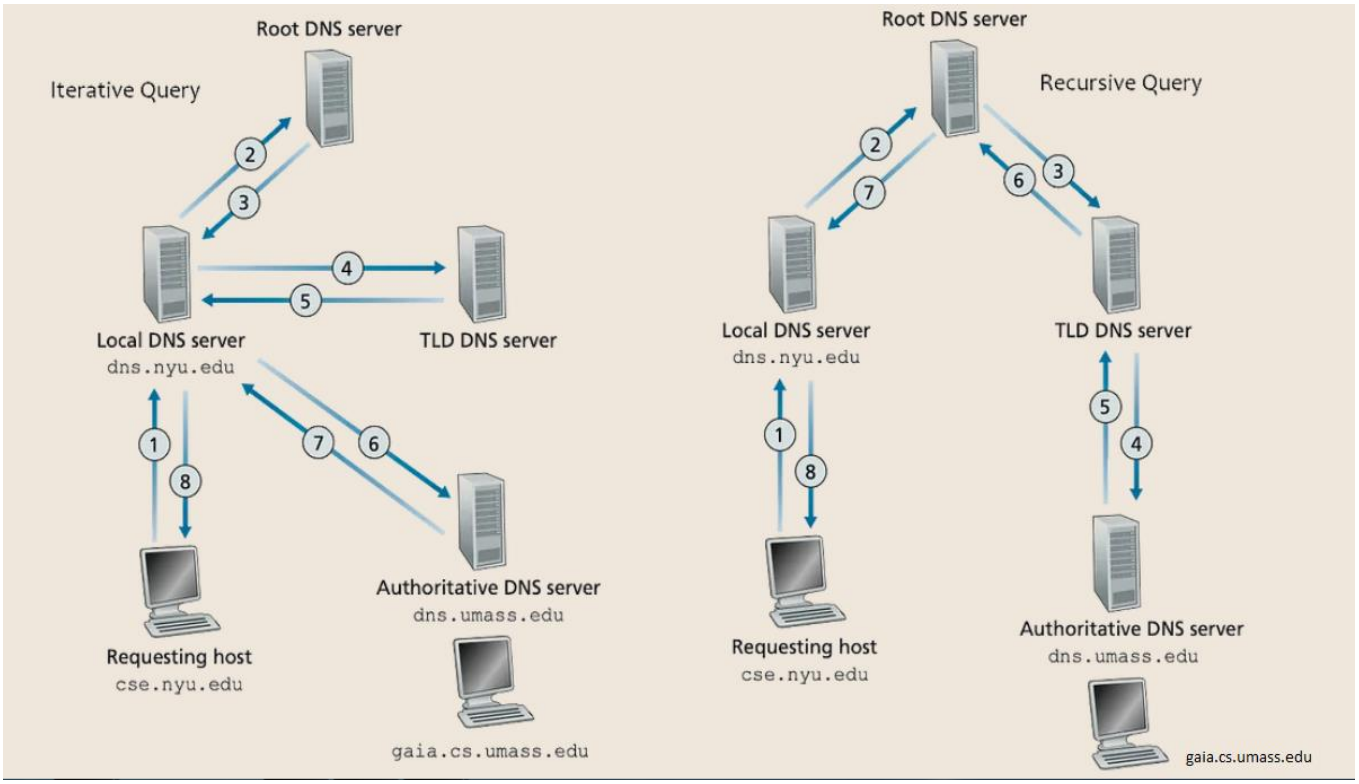
Figure: Portion of the hierarchy of DNS servers

- Suppose a client wants to determine IP address for host name "www.amazon.com":

1) The client first contacts one of the root servers, which returns IP addresses for TLD servers
2) Then, the client contacts one of these TLD servers.
   - The TLD server returns the IP address of an authoritative-server for "amazon.com".
3) Finally, the client contacts one of the authoritative-servers for amazon.com.
   - The authoritative-server returns the IP address for the host name "www.amazon.com".

Interaction of various DNS servers:

- **Recursive Queries & Iterative Queries**
- Figure 1: Recursive, Figure 2: Iterative



Recursion:

Step 1: Host at cis.poly.edu wants IP address for gaia.cs.umass.edu.Host sends "recursion requested" query to dns.poly.edu (local DNS server).
Step 2: Local DNS server forwards the query to Root DNS server.
Step 3: Root DNS server returns the IP address of Top Level Domain DNS server.
Step 4: Local DNS server contacts the Top Level Domain DNS server.
Step 5: Top Level Domain DNS server returns the IP address of Authoritative DNS server.
Step 6: Local DNS server contacts the Authoritative DNS server.
Step 7: Authoritative DNS server gives the final answer that is IP address of the gaia.cs.umass.edu. to the Local DNS server.
Step 8: Local DNS server forwards the answer to the requested host.

Iterative:

Step 1: Host at cis.poly.edu wants IP address for gaia.cs.umass.edu.Host sends "recursion requested" query to dns.poly.edu (local DNS server).
Step 2: Local DNS server forwards the query to Root DNS server.
Step 3: Root DNS server forwards the query to Top Level Domain DNS server.
Step 4: Top Level Domain DNS server forwards the query to Authoritative DNS server.
Step 5: Authoritative DNS server gives the final answer that is IP address of the gaia.cs.umass.edu. to the Top Level Domain DNS server.
Step 6: Top Level Domain DNS server forwards the answer to Root DNS server.
Step 7: Root DNS server forwards the answer to Local DNS server.
Step 8: Local DNS server forwards the answer to the requested host.

DNS Records & Messages

- The DNS servers contain the resource-records (RRs).
- RRs provide host name-to-IP address mappings.
- Each DNS reply message carries one or more resource-records.
- A resource-record is a 4-tuple that contains the following fields: (Name,Value,Type,TTL)
- TTL (time to live) determines when a resource should be removed from a cache.
- The meaning of Name and Value depend on Type:

1) If Type=A,then Name is a host name and Value is the IP address for the host name.
   - Thus, a Type A record provides the standard host name-to-IP address mapping.
   - For ex: (relay1.bar.foo.com, 145.37.93.126, A)

2) If Type=NS, then

i) Name is a domain (suchasfoo.com) and
ii) Value is the host name of an authoritative DNS server.

   - This record is used to route DNS queries further along in the query chain.
   - For ex:(foo.com, dns.foo.com, NS) is a Type NS record.

3) If Type=CNAME, then Value is a canonical host name for the alias host name Name.
   - This record can provide querying hosts the canonical name for a hostname.
   - For ex:(foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.

4) If Type=MX, Value is the canonical name of a mail-server that has an alias hostname Name.
   - MX records allow the host names of mail-servers to have simple aliases.
   - For ex:(foo.com, mail.bar.foo.com, MX) is an MX record.

DNS Messages

- Two types of DNS messages:1) query and 2)reply.
- Both query and reply messages have the same format.

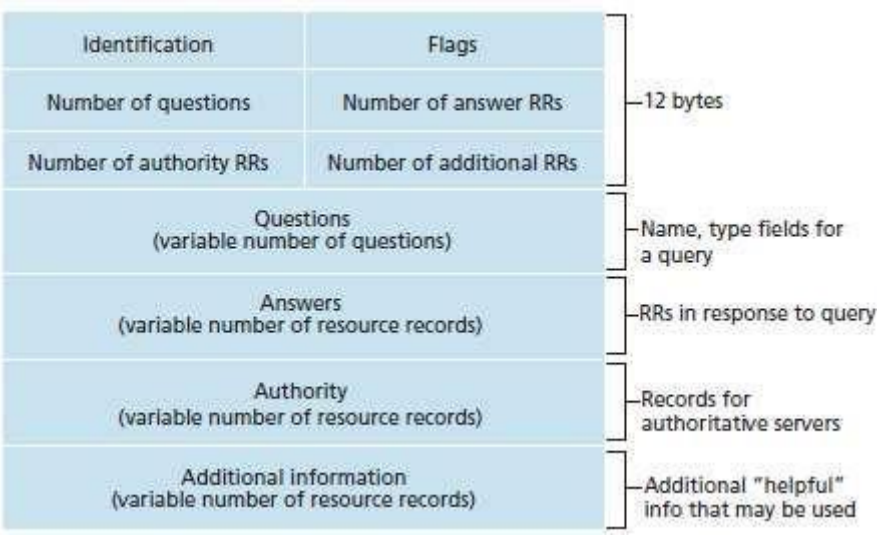


Figure:DNS message format

- The various fields in a DNS message are as follows:

1) Header Section

- The first 12 bytes is the header-section.
- This section has following fields:

i) Identification

- This field identifies the query.
- This identifier is copied in to the reply message to a query.
- This identifier allows the client to match received replies with sent queries.

ii) Flag

- This field has following 3flag-bits:

a) Query/Reply

¤This flag-bit indicates whether the message is a query(0) or a reply(1).

b) Authoritative

¤This flag-bit is set in a reply message when a DNS server is an authoritative-server.

c) Recursion Desired

¤This flag- bit is set when a client desires that the DNS server perform recursion.

iii) Four Number-of-Fields

- These fields indicate the no. of occurrences of 4 types of data sections that follow the header.

2) Question Section

- This section contains information about the query that is being made.
- This section has following fields:

i) Name

- This field contains the domain-name that is being queried.

ii) Type

- This field indicates the type of question being asked about the domain-name.

3) Answer Section

- This section contains a reply from a DNS server.
- This section contains the resource-records for the name that was originally queried.
- A reply can return multiple RRs in the answer, since a hostname can have multiple IP addresses.

4) Authority Section

- This section contains records of other authoritative-servers.

5) Additional Section

- This section contains other helpful records.