UNIT –I

Overview of the Internet

Introduction To Computer Networks

Modern world scenario is ever changing. Data Communication and network have changed the way business and other daily affair works. Now, they highly rely on computer networks and internetwork.

A set of devices often mentioned as nodes connected by media link is called a Network.

A node can be a device which is capable of sending or receiving data generated by other nodes on the network like a computer, printer etc. These links connecting the devices are called **Communication channels**.

Computer network is a telecommunication channel using which we can share data with other coomputers or devices, connected to the same network. It is also called Data Network. The best example of computer network is Internet.

Computer network does not mean a system with one Control Unit connected to multiple other systems as its slave. That is Distributed system, not Computer Network.

A network must be able to meet certain criterias, these are mentioned below:

1. Performance
2. Reliability
3. Scalability

Computer Networks: Performance

It can be measured in the following ways:

- **Transit time :** It is the time taken to travel a message from one device to another.
- **Response time :** It is defined as the time elapsed between enquiry and response.

Types of area networks – LAN, MAN and WAN

The **Network** allows computers to **connect and communicate** with different computers via any medium. LAN, MAN and WAN are the three major types of the network designed to operate over the area they cover. There are some similarities and dissimilarities between them. One of the major differences is the geographical area they cover, i.e.

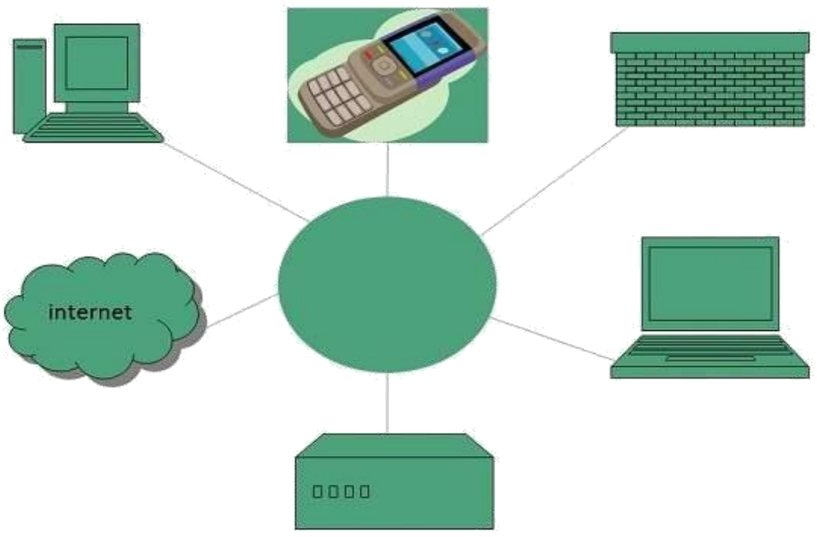
- **LAN** covers the smallest area;
- **MAN** covers an area larger than LAN
- **WAN** comprises the largest of all.

Internet Overview

Internet

Internet is defined as an Information super Highway, to access information over the web. However, It can be defined in many ways as follows:

- Internet is a world-wide global system of interconnected computer networks.
- Internet uses the standard Internet Protocol (TCP/IP).
- Every computer in internet is identified by a unique IP address.
- IP Address is a unique set of numbers (such as 110.22.33.114) which identifies a computer location.
- A special computer DNS (Domain Name Server) is used to give name to the IP Address so that user can locate a computer by a name.
- For example, a DNS server will resolve a name**http://www.gmail.com** to a particular IP address to uniquely identify the computer on which this website is hosted.
- Internet is accessible to every user all over the world.



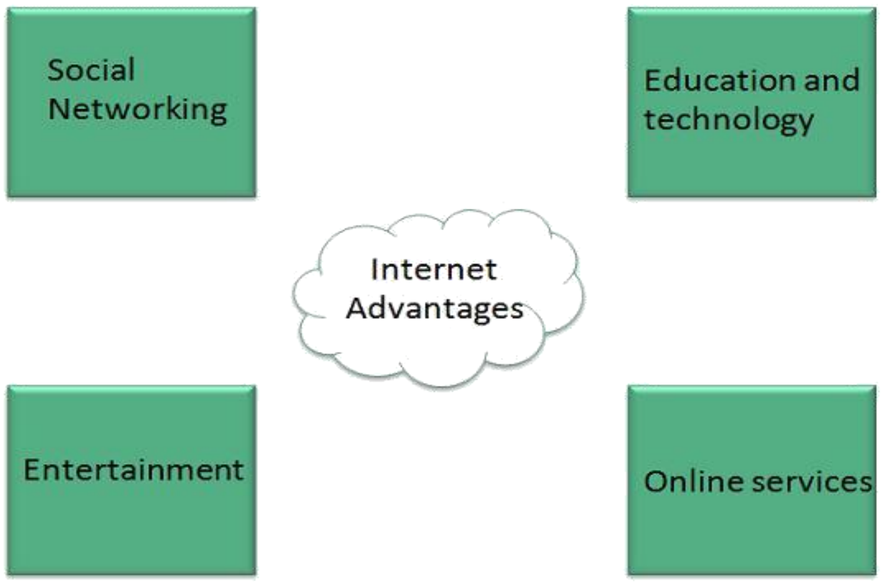
Evolution of Internet

The concept of Internet was originated in 1969 and has undergone several technological & Infrastructural changes as discussed below:

- The origin of Internet devised from the concept of**Advanced Research Project Agency Network (ARPANET).**
- **ARPANET** was developed by United States Department of Defense.
- Basic purpose of ARPANET was to provide communication among the various bodies of government.
- Initially, there were only four nodes, formally called **Hosts.**
- In 1972, the **ARPANET** spread over the globe with 23 nodes located at different countries and thus became known as **Internet.**
- By the time, with invention of new technologies such as TCP/IP protocols, DNS, WWW, browsers, scripting languages etc.,Internet provided a medium to publish and access information over the web.

Advantages

Internet covers almost every aspect of life, one can think of. Here, we will discuss some of the advantages of Internet:

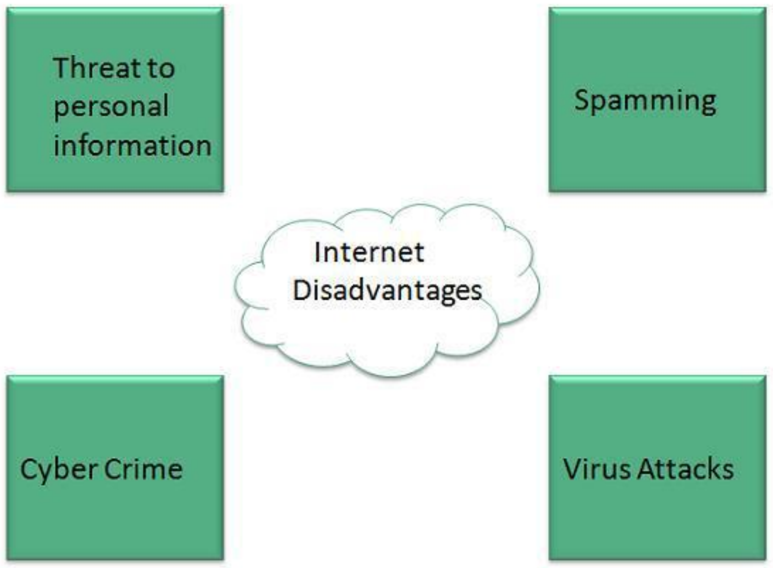


- Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the wed that uses Internet as a medium for communication. One can find various social networking sites such as:
    - Facebook
    - Twitter
    - Yahoo
    - Google+
    - Flickr
    - Orkut
- One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.
- Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet.
    - Online Television

    - Online Games
    - Songs
    - Videos
    - Social Networking Apps
- Internet allows us to use many services like:
    - Internet Banking
    - Matrimonial Services
    - Online Shopping
    - Online Ticket Booking
    - Online Bill Payment
    - Data Sharing
    - E-mail
- Internet provides concept of **electronic commerce**, that allows the business deals to be conducted on electronic systems

Disadvantages

However, Internet has prooved to be a powerful source of information in almost every field, yet there exists many disadvanatges discussed below:



- There are always chances to loose personal information such as name, address, credit card number. Therefore, one should be very careful while sharing such information. One should use credit cards only through authenticated sites.

- Another disadvantage is the **Spamming**.Spamming corresponds to the unwanted e-mails in bulk. These e-mails serve no purpose and lead to obstruction of entire system.
- **Virus** can easily be spread to the computers connected to internet. Such virus attacks may cause your system to crash or your important data may get deleted.
- Also a biggest threat on internet is pornography. There are many pornographic sites that can be found, letting your children to use internet which indirectly affects the children healthy mental life.
- There are various websites that do not provide the authenticated information. This leads to misconception among many people.

PROTOCOLS

In computer networks, communication occurs between entities in different systems. An entity is anything capable of sending or receiving information. However, two entities cannot simply send bit streams to each other and expect to be understood. For communication to occur, the entities must agree on a protocol. A protocol is a set of rules that govern data communications. A protocol defines what is communicated, how it is communicated, and when it is communicated. The key elements of a protocol are syntax, semantics, and timing.

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

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

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

Standards

Standards are essential in creating and maintaining an open and competitive market for equipment manufacturers and in guaranteeing national and international interoperability of data and telecommunications technology and processes. Standards provide guidelines to manufacturers, vendors, government agencies, and other service providers to ensure the kind of interconnectivity necessary in today's marketplace and in international communications.

Data communication standards fall into two categories: *de facto* (meaning "by fact" or "by convention") and *de jure* (meaning "by law" or "by regulation").

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

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

Standards are developed by cooperation among standards creation committees, forums, and government regulatory agencies.

Standards Creation Committees:

a) International Standards Organization (ISO)

b) International Telecommunications Union (ITU)

c) American National Standards Institute (ANSI)

d) Institute of Electrical and Electronics Engineers (IEEE)

e) Electronic Industries Association (EIA)

a) **International Standards Organization (ISO)**

A multinational body whose membership is drawn mainly from the standards creation committees of various governments throughout the world. Dedicated to worldwide agreement on international standards in a variety field. Currently includes 82 memberships industrialized nations. Aims to facilitate the international exchange of goods and services by providing models for compatibility, improved quality, increased quality, increased productivity and decreased prices.

b) International Telecommunications Union (ITU)

Also known as International Telecommunications Union-Telecommunication Standards Sector (ITU-T). An international standards organization related to the United Nations that develops standards for telecommunications.

c) American National Standards Institute (ANSI)

A non-profit corporation not affiliated with US government. ANSI members include professional societies, industry associations, governmental and regulatory bodies, and consumer groups. Discussing the internetwork planning and engineering, ISDN services, signaling, and architecture and optical hierarchy.

d) Institute of Electrical and Electronics Engineers (IEEE)

The largest national professional group involved in developing standards for computing, communication, electrical engineering, and electronics. Aims to advance theory, creativity and product quality in the fields of electrical engineering, electronics and radio. It sponsored an important standard for local area networks called Project 802 (eg. 802.3, 802.4 and 802.5 standards.)

e) Electronic Industries Association (EIA)

An association of electronics manufacturers in the US. Provide activities include public awareness education and lobbying efforts in addition to standards development. Responsible for developing the EIA-232-D and EIA-530 standards.

INTERNET STANDARDS

An Internet standard is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status. A specification begins as an Internet draft. An Internet draft is a working document (a work in progress) with no official status and a six-month lifetime. Upon recommendation from the Internet authorities, a draft may be published as a Request for Comment (RFC). Each RFC is edited, assigned a number, and made available to all interested parties. RFCs go through maturity levels and are categorized according to their requirement level.

The OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 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 (Day and Zimmermann, 1983). It was revised in 1995(Day, 1995). The model is called the ISO-OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems.

The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.

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

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

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

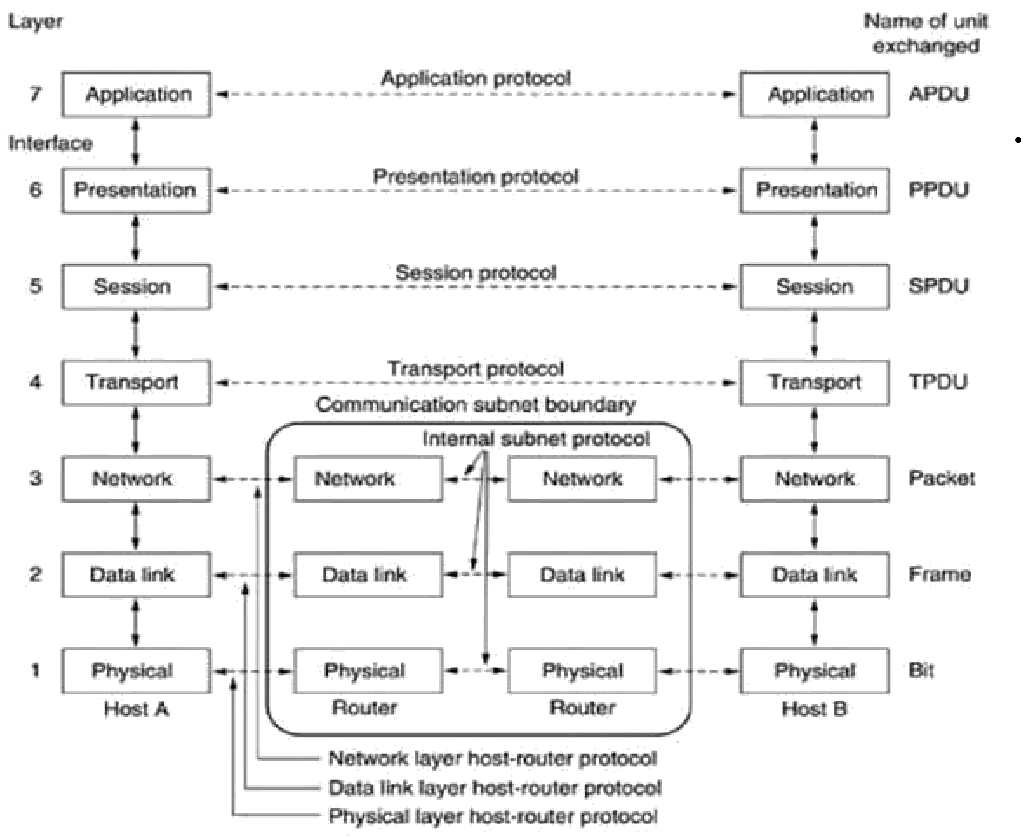


Fig.4: The OSI reference model

The Physical Layer:

The physical layer is concerned with transmitting raw bits over a communication channel. The design issues have to do with making 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.

The Data Link Layer:

The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer. It accomplishes this task by having the sender break up the input data into data frames (typically a few hundred or a few thousand bytes) and transmits the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgement frame.

Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism is often needed to let the transmitter know how much buffer space the receiver has at the moment. Frequently, this flow regulation and the error handling are integrated.

The Network Layer:

The network layer controls the operation of the subnet. A key design issue is 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. They can also be determined at the start of each conversation, for example, a terminal session (e.g., a login to a remote machine). Finally, they can be highly dynamic, being determined anew for each packet, to reflect the current network load.

If too many packets are present in the subnet at the same time, they will get in one another's way, forming bottlenecks. The control of such congestion also belongs to the network layer. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue.

When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from 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. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected. In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

The Transport Layer:

The basic function of the transport layer is to accept data from above, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology. The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service are the transporting of isolated messages, with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established.

The transport layer is a true end-to-end layer, all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers,

the protocols are between each machine and its immediate neighbours, and not between the ultimate source and destination machines, which may be separated by many routers.

The Session Layer:

The session layer allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management (preventing two parties from attempting the same critical operation at the same time), and synchronization (check pointing long transmissions to allow them to continue from where they were after a crash).

The Presentation Layer:

The presentation layer is concerned with the syntax and semantics of the information transmitted. In order to make it possible for computers with different data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire." The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records), to be defined and exchanged.

The Application Layer:

The application layer contains a variety of protocols that are commonly needed by users. One widely-used application protocol is HTTP (Hypertext Transfer Protocol), which 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 using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news.

The TCP/IP Reference Model

The TCP/IP reference model was developed prior to OSI model. The major design goals of this model were,

1. To connect multiple networks together so that they appear as a single network.
2. To survive after partial subnet hardware failures.
3. To provide a flexible architecture.

Unlike OSI reference model, TCP/IP reference model has only 4 layers. They are,

1. Host-to-Network Layer
2. Internet Layer

3. Transport Layer

4. Application Layer

Host-to-Network Layer:

The TCP/IP reference model does not really say much about what happens here, except to point out that the host has to connect to the network using some protocol so it can send IP packets to it. This protocol is not defined and varies from host to host and network to network.

Internet Layer:

This layer, called the internet layer, is the linchpin that holds the whole architecture together. Its job is to permit hosts to inject packets into any network and have they travel independently to the destination (potentially on a different network). They may even arrive in a different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired. Note that "internet" is used here in a generic sense, even though this layer is present in the Internet.

The internet layer defines an official packet format and protocol called IP (Internet Protocol). The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly the major issue here, as is avoiding congestion. For these reasons, it is reasonable to say that the TCP/IP internet layer is similar in functionality to the OSI network layer. Fig. shows this correspondence.

The Transport Layer:

The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here. The first one, TCP (Transmission Control Protocol), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet. It fragments the incoming byte stream into discrete messages and passes each one on to the internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

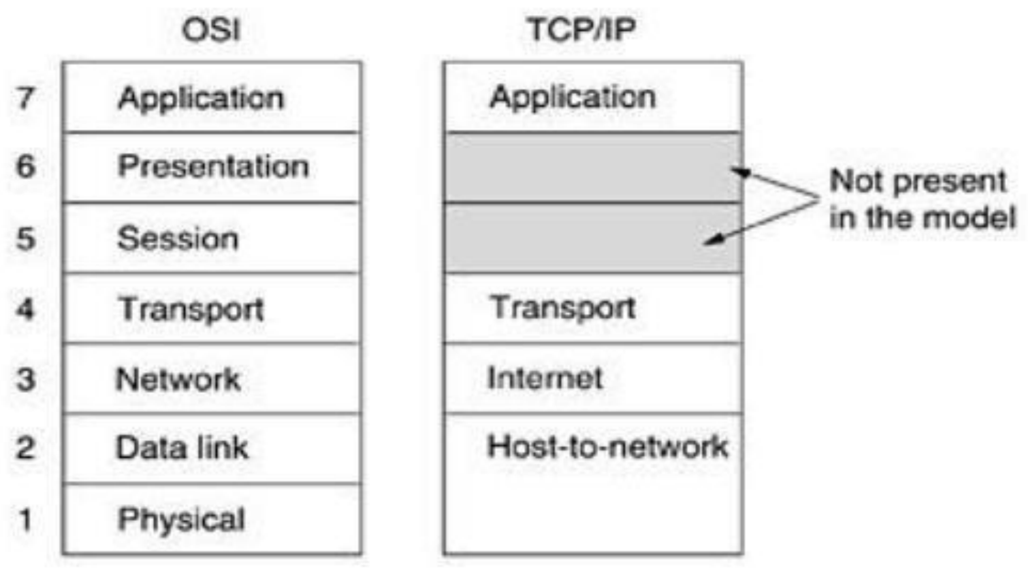


Fig.1: The TCP/IP reference model.

The second protocol in this layer, UDP (User Datagram Protocol), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video. The relation of IP, TCP, and UDP is shown in Fig.2. Since the model was developed, IP has been implemented on many other networks.

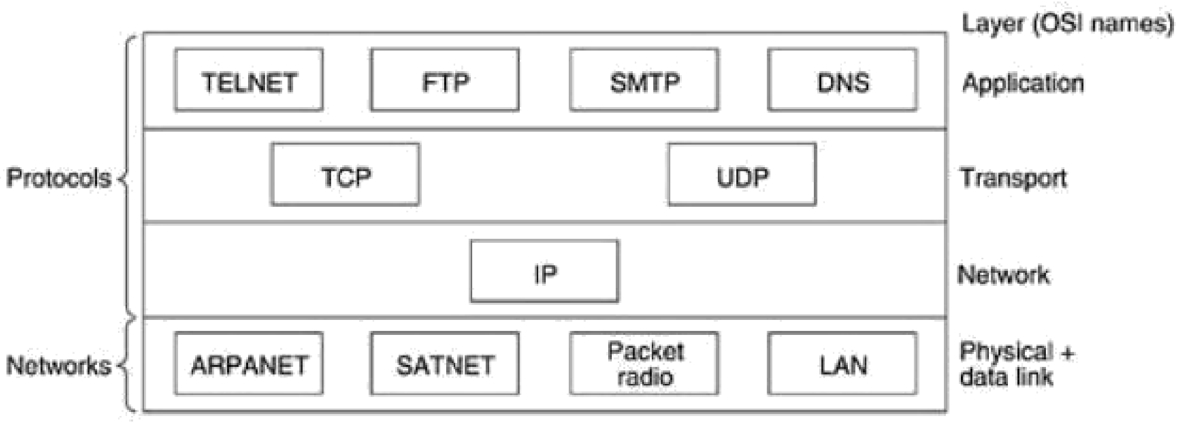


Fig.2: Protocols and networks in the TCP/IP model initially

The Application Layer:

The TCP/IP model does not have session or presentation layers. On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP), as shown in Fig.6.2. The virtual terminal protocol allows a user on one machine to log onto a distant machine and work there. The file transfer protocol provides a way to move data efficiently from one machine to another. Electronic mail was originally just a kind of file transfer, but later a specialized protocol (SMTP) was developed for it. Many other protocols have been added to these over the years: the Domain Name System (DNS) for mapping host names onto their network addresses, NNTP, the protocol for moving USENET news articles around, and HTTP, the protocol for fetching pages on the World Wide Web, and many others.

Comparison of the OSI and TCP/IP Reference Models

Now it's time to compare both the reference model that we have learned till now. Let's start by addressing the similarities that both of these models have.

Following are some **similarities** between OSI Reference Model and TCP/IP Reference Model.

- Both have layered architecture.
- Layers provide similar functionalities.

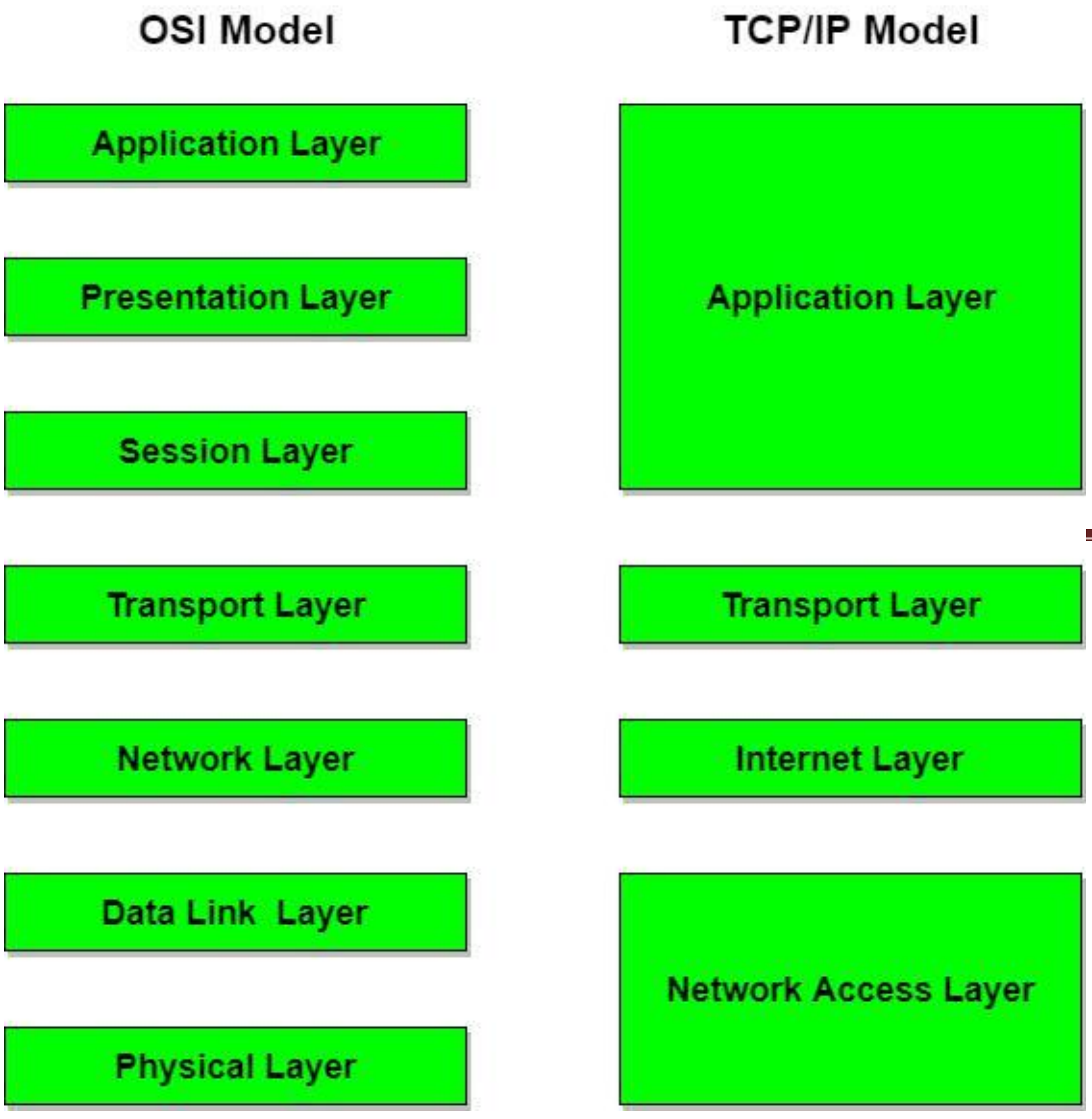
Difference between OSI and TCP/IP Reference Model

Following are some major differences between OSI Reference Model and TCP/IP Reference Model, with diagrammatic comparison below.

OSI(Open System Interconnection)	**TCP/IP(Transmission Control Protocol / Internet Protocol)**
1. OSI is a generic, protocol independent standard, acting as a communication gateway between the network and end user.	1. TCP/IP model is based on standard protocols around which the Internet has developed. It is a communication protocol, which allows connection of hosts over a network.
2. In OSI model the transport layer guarantees the delivery of packets.	2. In TCP/IP model the transport layer does not guarantees delivery of packets. Still the TCP/IP model is more reliable.
3. Follows vertical approach.	3. Follows horizontal approach.
4. OSI model has a separate Presentation layer and Session layer.	4. TCP/IP does not have a separate Presentation layer or Session layer.
5. Transport Layer is Connection Oriented.	5. Transport Layer is both Connection Oriented and Connection less.

	of the OSI model.
6. Network Layer is both Connection Oriented and Connection less.	6. Network Layer is Connection less.
7. OSI is a reference model around which the networks are built. Generally it is used as a guidance tool.	7. TCP/IP model is, in a way implementation
8. Network layer of OSI model provides both connection oriented and connectionless service.	8. The Network layer in TCP/IP model provides connectionless service.
9. OSI model has a problem of fitting the protocols into the model.	9. TCP/IP model does not fit any protocol
10. Protocols are hidden in OSI model and are easily replaced as the technology changes.	10. In TCP/IP replacing protocol is not easy.
11. OSI model defines services, interfaces and protocols very clearly and makes clear distinction between them. It is protocol independent.	11. In TCP/IP, services, interfaces and protocols are not clearly separated. It is also protocol dependent.

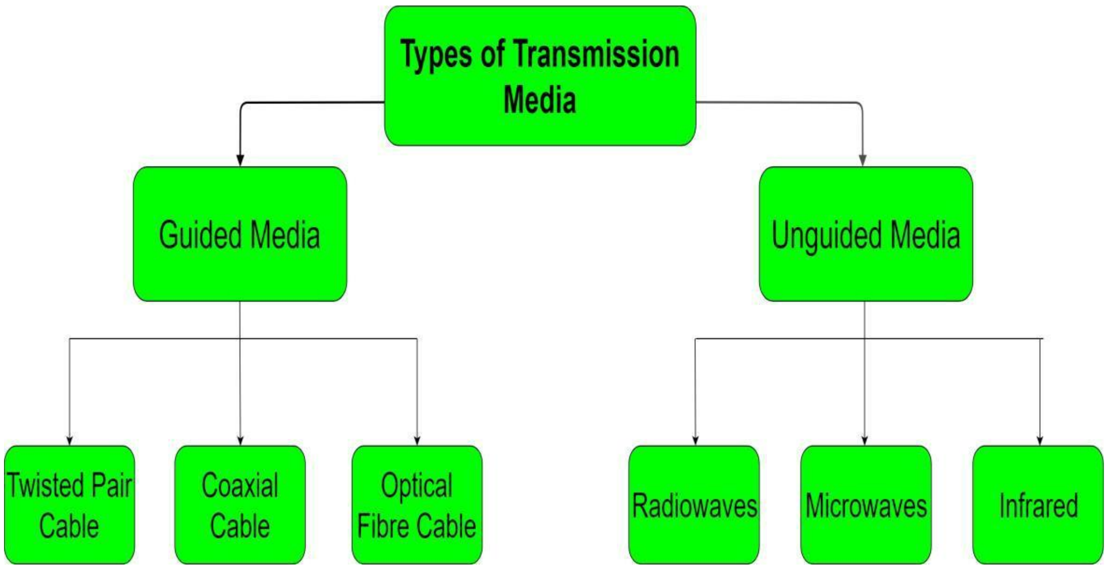
Diagrammatic Comparison between OSI Reference Model and TCP/IP Reference Model



Physical Layer

TRANSMISSION MEDIA

In data communication terminology, a transmission medium is a physical path between the transmitter and the receiver i.e it is the channel through which data is sent from one place to another. Transmission Media is broadly classified into the following types:



A transmission **medium** can be broadly defined as anything that can carry information from a source to a destination. For example, the transmission medium for two people having a dinner conversation is the air. The air can also be used to convey the message in a smoke signal or semaphore. For a written message, the transmission medium might be a mail carrier, a truck, or an airplane.

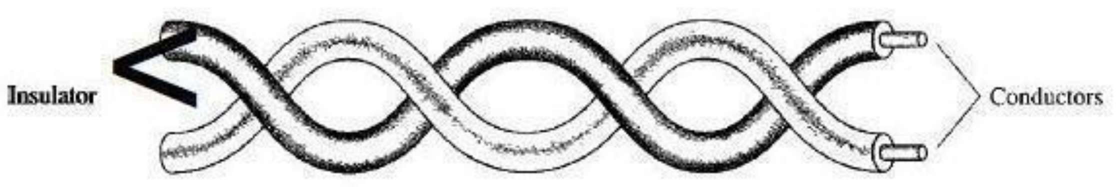
In data communications the definition of the information and the transmission medium is more specific. The transmission medium is usually free space, metallic cable or optical cable. The information is usually a signal that is the result of conversion of data from another form.

I.Guided Media

Guided media, which are those that provide a conduit from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light.

1. <u>**Twisted-Pair Cable**</u>

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



One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals. If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e,g., one is closer and the other is farther). This results in a difference at the receiver. By twisting the pairs, a balance is maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true. Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals. The unwanted signals are mostly canceled out. From the above discussion, it is clear that the number of twists per unit of length (e.g., inch) has some effect on the quality of the cable.

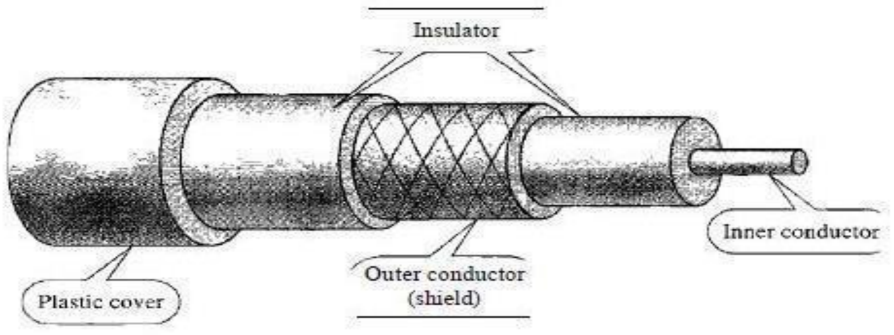
Applications

Twisted-pair cables are used in telephone lines to provide voice and data channels. The local loop-the line that connects subscribers to the central telephone office-commonly consists of

Unshielded twisted pair cables. The DSL line that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables. Local-area networks, such as lOBase-T and lOOBase-T, also use twisted-pair cables.

2.**Coaxial Cable**

Coaxial cable (or *coax)* carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover (below figure).



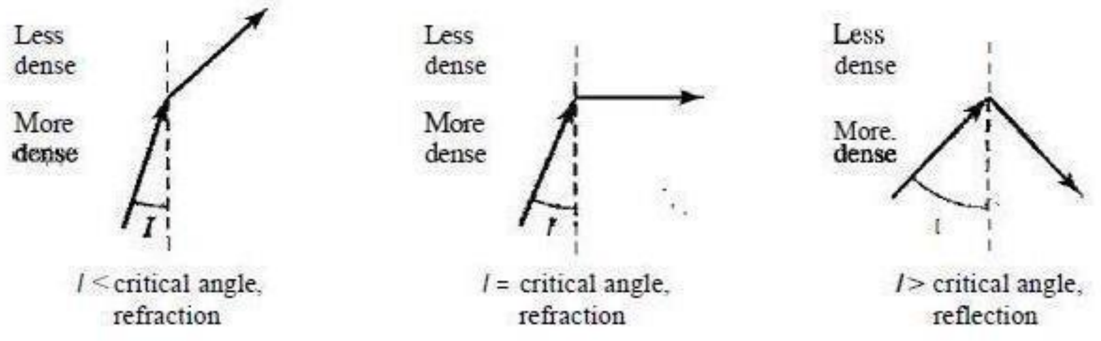
Applications

Coaxial cable was widely used in analog telephone networks where a single coaxial network could carry 10,000 voice signals. Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps. However, coaxial cable in telephone networks has largely been replaced today with fiber-optic cable. Cable TV networks also use coaxial cables.

In the traditional cable TV network, the entire network used coaxial cable. Later, however, cable TV providers replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable. Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs.

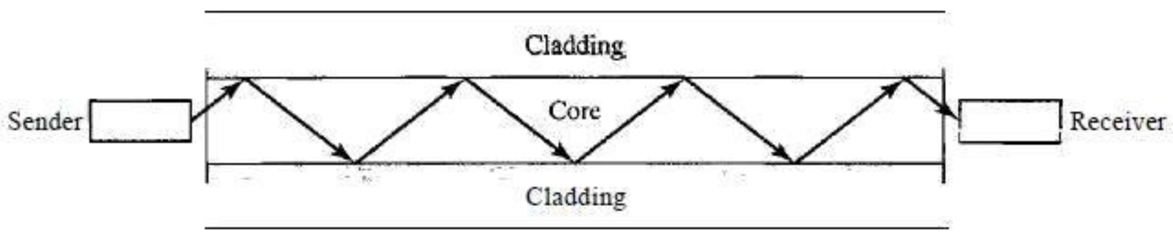
3. Fiber Optic Cable

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light. Light travels in a straight line as long as it is moving through a single uniform If a ray of light traveling through one substance suddenly enters another substance (of a different density), the ray changes direction. Figure 7.10 shows how a ray of light changes direction when going from a more dense to a less dense substance.


Less dense
More dense
I
I < critical angle, refraction
Less dense
More dense
I
I = critical angle, refraction
Less dense
More dense
I
I > critical angle, reflection

As the figure shows, if the angle of incidence *I* (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the critical angle, the ray refracts and moves closer to the surface. If the angle of incidence is equal to the critical angle, the light bends along the interface. If the angle is greater than the critical angle, the ray reflects (makes a turn) and travels again in the denser substance. Note that the critical angle is a property of the substance, and its value differs from one substance to another.

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



Applications

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective. Today, with wavelength-division multiplexing (WDM), we can transfer data at a rate of 1600 Gbps. The SONET network provides such a backbone. Some cable TV companies use a combination of optical fiber and coaxial cable, thus creating a hybrid network. Optical fiber provides the backbone structure while coaxial cable provides the connection to the user premises. This is a cost-effective configuration since the narrow bandwidth requirement at the user end does not justify the use of optical fiber. Local-area networks such as 100Base-FX network (Fast Ethernet) and 1000Base-X also use fiber-optic cable.

Advantages and Disadvantages of Optical Fiber Advantages

Fiber-optic cable has several advantages over metallic cable (twisted pair or coaxial).

1. **Higher bandwidth**. Fiber-optic cable can support dramatically higher bandwidths (and hence data rates) than either twisted-pair or coaxial cable. Currently, data rates and bandwidth utilization over fiber-optic cable are limited not by the medium but by the signal generation and reception technology available.

2. **Less signal attenuation**. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.

3. **Immunity to electromagnetic interference**. Electromagnetic noise cannot affect fiber-optic cables.

4. **Resistance to corrosive materials**. Glass is more resistant to corrosive materials than copper.

5. **Light weight**. Fiber-optic cables are much lighter than copper cables.

6. **Greater immunity to tapping**. Fiber-optic cables are more immune to tapping than copper cables. Copper cables create antenna effects that can easily be tapped.

Disadvantages

There are some disadvantages in the use of optical fiber.

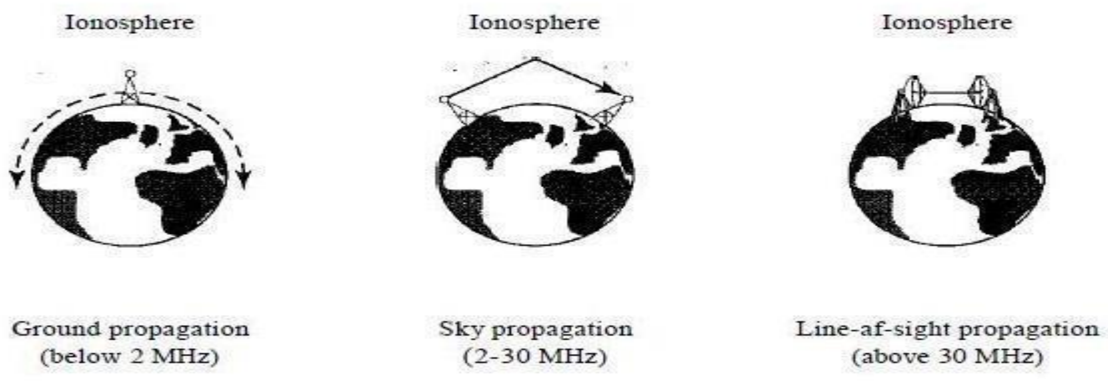
1. **Installation and maintenance**. Fiber-optic cable is a relatively new technology. Its installation and maintenance require expertise that is not yet available everywhere.

2. **Unidirectional light propagation**. Propagation of light is unidirectional. If we need bidirectional communication, two fibers are needed.

3. **Cost**. The cable and the interfaces are relatively more expensive than those of other guided media. If the demand for bandwidth is not high, often the use of optical fiber cannot be justified.

II. UNGUIDED MEDIA: WIRELESS

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication. Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them.



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

Figure 7.18. In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth.

These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance. In sky propagation, higher-frequency radio waves radiate upward into the ionosphere where they are reflected back to earth. This type of transmission allows for greater distances with lower output power.

In line of sight propagation, very high frequency signals are transmitted in straight lines directly from antenna to antenna. Antennas must be directional, facing each other, and either tall enough or close enough together not to be affected by the curvature of the earth. Line-of-sight propagation is tricky because radio transmissions cannot be completely focused.

1. Radio Waves

Waves ranging in frequencies between 3 kHz and 1 GHz are called radio waves. Radio waves, for the most part, are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna. The omnidirectional property has a disadvantage, too.

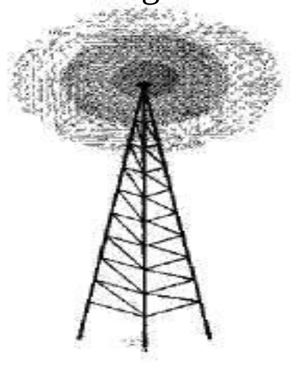
The radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band. Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting

such as AM radio. Radio waves, particularly those of low and medium frequencies, can penetrate walls.

This characteristic can be both an advantage and a disadvantage. It is an advantage because, for example, an AM radio can receive signals inside a building. It is a disadvantage because we cannot isolate a communication to just inside or outside a building. The radio wave band is relatively narrow, just under 1 GHz, compared to the microwave band. When this band is divided into sub bands, the sub bands are also narrow, leading to a low data rate for digital communications.

Omnidirectional Antenna

Radio waves use omnidirectional antennas that send out signals in all directions. Based on the wavelength, strength, and the purpose of transmission, we can have several types of antennas. Below figure 7.20 shows an omnidirectional antenna.



Applications

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

2. Microwaves

Electromagnetic waves having frequencies between I and 300 GHz are called microwaves. Microwaves are unidirectional. When an antenna transmits microwave waves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas. The following describes some characteristics of microwave propagation:

1. Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles do not allow two short towers to communicate by using microwaves. Repeaters are often needed for long distance communication.
2. Very high-frequency microwaves cannot penetrate walls. This

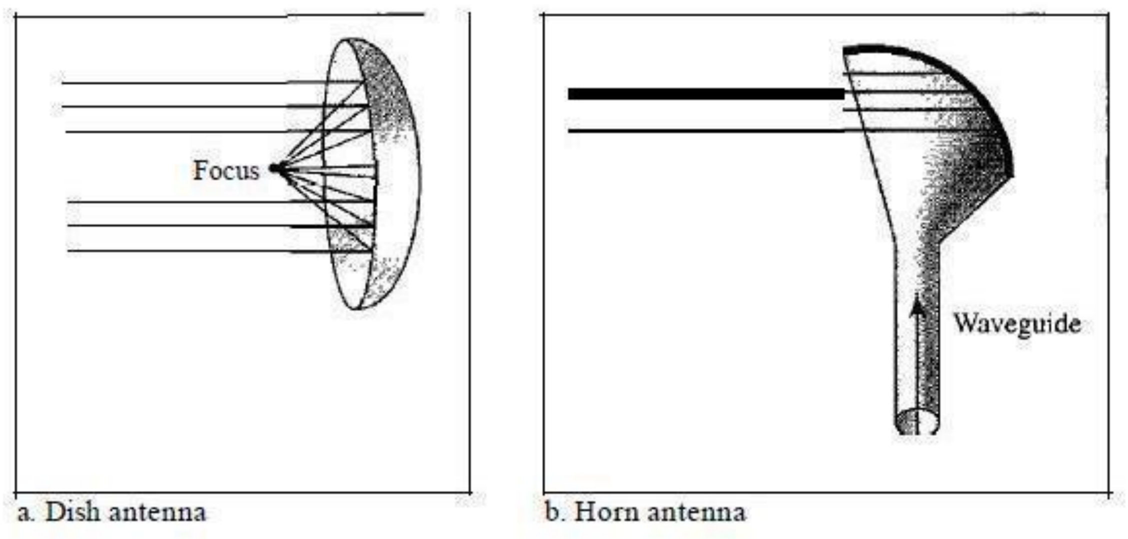
   characteristic can be a disadvantage if receivers are inside buildings.
3. The microwave band is relatively wide, almost 299 GHz. Therefore wider sub bands can be assigned, and a high data rate is possible.
4. Use of certain portions of the band requires permission from authorities.

Unidirectional Antenna

Microwaves need unidirectional antennas that send out signals in one direction. Two types of antennas are used for microwave communications: the parabolic dish and the horn (see below figure). A parabolic dish antenna is based on the geometry of a parabola: Every line parallel to the line of symmetry (line of sight) reflects off the curve at angles such that all the lines intersect in a common point called the focus.

The parabolic dish works as a funnel, catching a wide range of waves and directing them to a common point. In this way, more of the signal is recovered than would be possible with a single-point receiver. Outgoing transmissions are broadcast through a horn aimed at the dish.

The microwaves hit the dish and are deflected outward in a reversal of the receipt path. A horn antenna looks like a gigantic scoop. Outgoing transmissions are broadcast up a stem (resembling a handle) and deflected outward in a series of narrow parallel beams by the curved head. Received transmissions are collected by the scooped shape of the horn, in a manner similar to the parabolic dish, and are deflected down into the stem.



3. Infrared

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

Applications

The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate. The *Infrared Data Association* (IrDA), an association for sponsoring the use of infrared waves, has established standards for using these signals for communication between devices such as keyboards, mice, PCs, and printers. For example, some manufacturers provide a special port called the IrDA port that allows a wireless keyboard to communicate with a PC. The standard originally defined a data rate of 75 kbps for a distance up to 8 m. The recent standard defines a data rate of 4 Mbps.

Infrared signals defined by IrDA transmit through line of sight; the IrDA port on the keyboard needs to point to the PC for transmission to occur.

DATA LINK LAYER

Introduction

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

DATA LINK LAYER DESIGN ISSUES

The following are the data link layer design issues

1. Services Provided to the Network Layer

The network layer wants to be able to send packets to its neighbors without worrying about the details of getting it there in one piece.

2. Framing

Group the physical layer bit stream into units called frames. Frames are nothing more than "packets" or "messages". By convention, we use the term "frames" when discussing DLL.

3. Error Control

Sender checksums the frame and transmits checksum together with data. Receiver re-computes the checksum and compares it with the received value.

4. Flow Control

Prevent a fast sender from overwhelming a slower receiver.

<u>Services Provided to the Network Layer</u>

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

The data link layer can be designed to offer various services. The actual services offered can vary from system to system. Three reasonable possibilities that are commonly provided are

1) **Unacknowledged Connectionless service**

2) **Acknowledged Connectionless service**

3) **Acknowledged Connection-Oriented service**

In **Unacknowledged connectionless service** consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them.

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.

This class of service is appropriate when the error rate is very low so that recovery is left to higher layers. It is also appropriate for real-time traffic, such as voice, in which late data are worse than bad data. Most LANs use unacknowledged connectionless service in the data link layer.

When **Acknowledged Connectionless 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. 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.

Adding Ack in the DLL rather than in the Network Layer is just an optimization and not a requirement. If individual frames are acknowledged and retransmitted, entire packets get through much faster. On reliable channels, such as fiber, the overhead of a heavyweight data link protocol may be unnecessary, but on wireless channels, with their inherent unreliability, it is well worth the cost.

In Acknowledged Connection-Oriented service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

When connection-oriented service is used, transfers go through three distinct phases.

In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not.

In the second phase, one or more frames are actually transmitted.

In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection

CYCLIC CODES

The cyclic codes are special class of linear block codes which has property of generating a new code word when the given codeword is shifted cyclically. For e.g., if we assume the bits of first word as a0 to a6 and bits in the second word can be obtained by shifting as shown below.

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

CYCLIC REDUNCANCY CHECK (CRC)

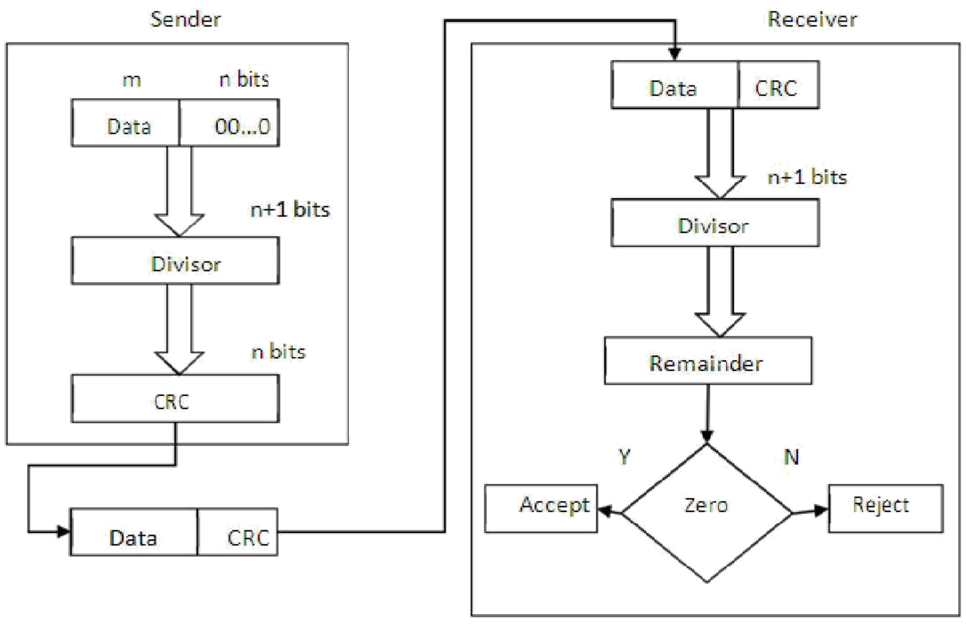
The cyclic redundancy check codes are popularly employed in LANs and WANs for error correction. The principle of operation of CRC encoders and decoders can be better explained with the following examples.

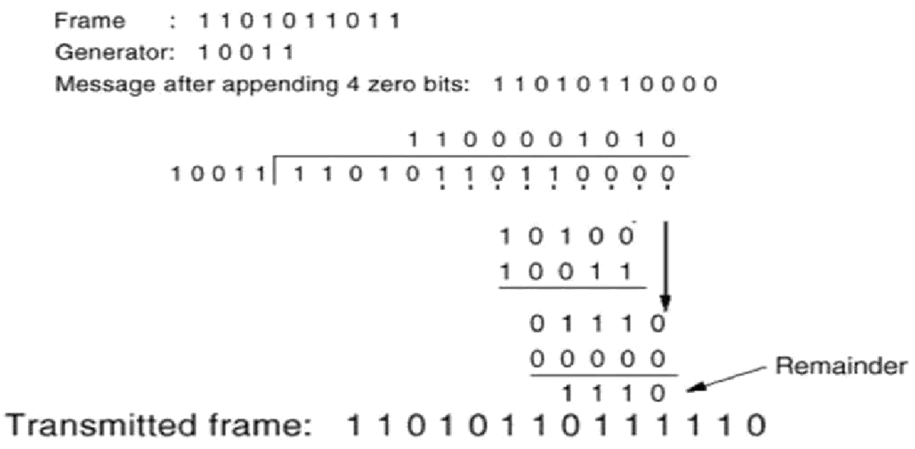
CRC is the most powerful and easy to implement technique.CRC is based on *binary division*. In CRC, a sequence of redundant bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.

At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected. The binary number, which is (r+1) bit in length, can also be considered as the coefficients of a polynomial, called *Generator Polynomial*.

PERFORMANCE OF CRC

CRC is a very effective error detection technique. If the divisor is chosen according to the previously mentioned rules, its performance can be summarized as follows. CRC can detect all single-bit errors and double bit errors (three 1‘s). CRC can detect any odd number of errors (X+1) and it can also detect all burst errors of less than the degree of the polynomial.




Frame : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after appending 4 zero bits: 1 1 0 1 0 1 1 0 0 0 0
1 1 0 0 0 0 1 0 1 0
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
1 0 1 0 0
1 0 0 1 1
0 1 1 1 0
0 0 0 0 0
1 1 1 0
Remainder
Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Elementary Data Link Protocols

An unrestricted simplex protocol

In order to appreciate the step by step development of efficient and complex protocols such as SDLC, HDLC etc., we will begin with a simple but unrealistic protocol. In this protocol:

- Data are transmitted in one direction only
- The transmitting (Tx) and receiving (Rx) hosts are always ready
- Processing time can be ignored
- Infinite buffer space is available
- No errors occur; i.e. no damaged frames and no lost frames (perfect channel)

The protocol consists of two procedures, a sender and receiver as depicted below:

```
/* protocol 1 */

Sender()
{
    forever
    {
        from_host(buffer);
        S.info = buffer;
        sendf(S);
    }
}

Receiver()
{
    forever
    {
        wait(event);
        getf(R);
        to_host(R.info);
    }
}
```

A simplex stop-and-wait protocol

In this protocol we assume that

- Data are transmitted in one direction only
- No errors occur (perfect channel)
- The receiver can only process the received information at a finite rate

These assumptions imply that the transmitter cannot send frames at a rate faster than the receiver can process them.

The problem here is how to prevent the sender from flooding the receiver.

A general solution to this problem is to have the receiver provide some sort of feedback to the sender. The process could be as follows: The receiver send an acknowledge frame back to the sender telling the sender that the last received frame has been processed and passed to the host; permission to send the next frame is granted. The sender, after having sent a frame, must wait for the acknowledge frame from the receiver before sending another frame. This protocol is known as *stop-and-wait*.

The protocol is as follows:

```
/* protocol 2 */

Sender()
{
    forever
    {
        from_host(buffer);
        S.info = buffer;
        sendf(S);
        wait(event);
    }
}

Receiver()
{
    forever
    {
        wait(event);
        getf(R);
        to_host(R.info);
        sendf(S);
    }
}
```

A simplex protocol for a noisy channel

In this protocol the unreal "error free" assumption in protocol 2 is dropped. Frames may be either damaged or lost completely. We assume that transmission errors in the frame are detected by the hardware checksum.

One suggestion is that the sender would send a frame, the receiver would send an ACK frame only if the frame is received correctly. If the frame is in error the receiver simply ignores it; the transmitter would time out and would retransmit it.

One fatal flaw with the above scheme is that if the ACK frame is lost or damaged, duplicate frames are accepted at the receiver without the receiver knowing it.

Imagine a situation where the receiver has just sent an ACK frame back to the sender saying that it correctly received and already passed a frame to its host. However, the ACK frame gets lost completely, the sender times out and retransmits the frame. There is no way for the receiver to tell whether this frame is a retransmitted frame or a new frame, so the receiver accepts this duplicate happily and transfers it to the host. The protocol thus fails in this aspect.

To overcome this problem it is required that the receiver be able to distinguish a frame that it is seeing for the first time from a retransmission. One way to achieve this is to have the sender put a sequence number in the header of each frame it sends. The receiver then can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.

The receiver needs to distinguish only 2 possibilities: a new frame or a duplicate; a 1-bit sequence number is sufficient. At any instant the receiver expects a particular sequence number. Any wrong sequence numbered frame arriving at the receiver is rejected as a duplicate. A correctly numbered frame arriving at the receiver is accepted, passed to the host, and the expected sequence number is incremented by 1 (modulo 2).

The protocol is depicted below:

```
/* protocol 3 */

Sender()
{
    NFTS = 0;            /* NFTS = Next Frame To Send */
    from_host(buffer);
    forever
    {
            S.seq = NFTS;
            S.info = buffer;
            sendf(S);
            start_timer(S.seq);
            wait(event);
            if(event == frame_arrival)
            {
                  from_host(buffer);
                  ++NFTS;  /* modulo 2 operation */
            }
      }
}

wait(event);
            if(event == frame_arrival)
            {
                  getf(R);
                  if(R.seq == FE)
                  {


                        to_host(R.info);
                        ++FE; /* modulo 2 operation */
                  }
                  sendf(S);     /* ACK */
            }
      }
}
```

This protocol can handle lost frames by timing out. The timeout interval has to be long enough to prevent premature timeouts which could cause a "deadlock" situation.

Sliding Window Protocols

Piggybacking technique

In most practical situations there is a need for transmitting data in both directions (i.e. between 2 computers). A full duplex circuit is required for the operation.

If protocol 2 or 3 is used in these situations the data frames and ACK (control) frames in the reverse direction have to be interleaved. This method is acceptable but not efficient. An efficient method is to absorb the ACK frame into the header of the data frame going in the same direction. This technique is known as*piggybacking*.

When a data frame arrives at an IMP (receiver or station), instead of immediately sending a separate ACK frame, the IMP restrains itself and waits until the host passes it the next message. The acknowledgement is then attached to the outgoing data frame using the ACK field in the frame header. In effect, the acknowledgement gets a free ride in the next outgoing data frame.

This technique makes better use of the channel bandwidth. The ACK field costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum.

An issue arising here is the time period that the IMP waits for a message onto which to piggyback the ACK. Obviously the IMP cannot wait forever and there is no way to tell exactly when the next message is available. For these reasons the waiting period is usually a fixed period. If a new host packet arrives quickly the acknowledgement is piggybacked onto it; otherwise, the IMP just sends a separate ACK frame.

Sliding window

When one host sends traffic to another it is desirable that the traffic should arrive in the same *sequence* as that in which it is dispatched. It is also desirable that a data link should deliver frames in the order sent.

A flexible concept of sequencing is referred to as the *sliding window* concept and the next three protocols are all sliding window protocols.

In all sliding window protocols, each outgoing frame contains a sequence number SN ranging from 0 to 2^(n -1)(where n is the number of bits reserved for the sequence number field).

At any instant of time the sender maintains a list of consecutive sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the *sending window.* Similarly, the receiver maintains a *receiving window* corresponding to frames it is permitted to accept.

The size of the window relates to the available buffers of a receiving or sending node at which frames may be arranged into sequence.

At the receiving node, any frame falling outside the window is discarded. Frames falling within the receiving window are accepted and arranged into sequence. Once sequenced, the frames at the left of the window are delivered to the host and an acknowledgement of the delivered frames is transmitted to their sender. The window is then rotated to the position where the left edge corresponds to the next expected frame, RN.

Whenever a new frame arrives from the host, it is given the next highest sequence number, and the upper edge of the sending window is advanced by one. The sequence numbers within the sender's window represent frames sent but as yet not acknowledged. When an acknowledgement comes in, it gives the position of the receiving left window edge which indicates what frame the receiver expects to receive next. The sender then rotates its window to this position, thus making buffers available for continuous transmission.

A one bit sliding window protocol: protocol 4

The sliding window protocol with a maximum window size 1 uses stop-and-wait since the sender transmits a frame and waits for its acknowledgement before sending the next one.

```
/* protocol 4 */

Send_and_receive()
{
    NFTS = 0;
    FE = 0;
    from_host(buffer);
    S.info = buffer;
    S.seq = NFTS;
    S.ack = 1-FE;
    sendf(S);
    start_timer(S.seq);
    forever
    {
        wait(event);
        if(event == frame_arrival)
        {
            getf(R);
            if(R.seq == FE)
            {
                to_host(R.info);
                ++FE;
            }
            if(R.ack == NFTS)
            {
                from_host(buffer);
                ++NFTS;
            }
        }
        S.info = buffer;
        S.seq = NFTS;
        S.ack = 1-FE;
        sendf(S);
        start_timer(S.seq);
    }
```

}

Pipelining

In many situations the long round-trip time can have important implications for the efficiency of the bandwidth utilisation.
As an example, consider a satellite channel with a 500ms round-trip propagation delay. At time t~~=0 the sender starts sending the first frame. Not until at least t~>=~500 ms has the acknowledgement arrived back at the sender. This means that the sender was blocked most of the time causing a reduction in efficiency.
As another example, if the link is operated in a two-way alternating mode (half-duplex), the line might have to be "turned around" for each frame in order to receive an acknowledgement. This acknowledgement delay could severely impact the effective data transfer rate.

The effects of these problems can be overcome by allowing the sender to transmit multiple contiguous frames (say up to w frames) before it receives an acknowledgement. This technique is known as *pipelining.*

In the satellite example, with a channel capacity of 50kbps and 1000-bit frames, by the time the sender has finished sending 26 frames, t~=~520 ms, the acknowledgement for frame 0 will have just arrived, allowing the sender to continue sending frames. At all times, 25 or 26 unacknowledged frames will be outstanding, and the sender's window size needs to be at least 26.

Pipelining frames over an unreliable communication channel raises some serious issues. What happens if a frame in the middle of a long stream is damaged or lost? What should the receiver do with all the correct frames following the bad one?

The are two basic *Automatic Request for Repeat (ARQ)* methods for dealing with errors in the presence of pipelining.

One method, the normal mode of ARQ is called *Go-back-N*. If the receiver detects any error in frame N, it signals the sender and then discards any subsequent frame.

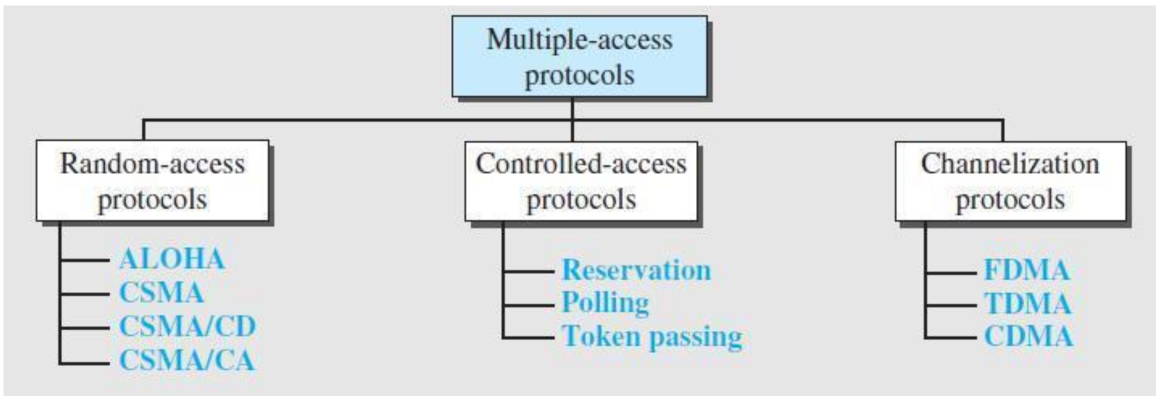
The sender, which may currently be sending frame N+X when the error signal is detected, initiates retransmission of frame N and all subsequent frames.

The other method is called *selective reject.* In this method the receiver stores all the correct frames following the bad one. When the sender finally notices what was wrong, it just retransmits the one bad frame, not all its successors.

UNIT-II

MULTIPLE ACCESS PROTOCOLS

The multiple access protocols can be broadly classified into three categories namely Random access Protocols, Controlled access Protocols and Channelization Protocols (as given in below figure). Let us discuss in detail about the different protocols which are classified and as shown in below figure.



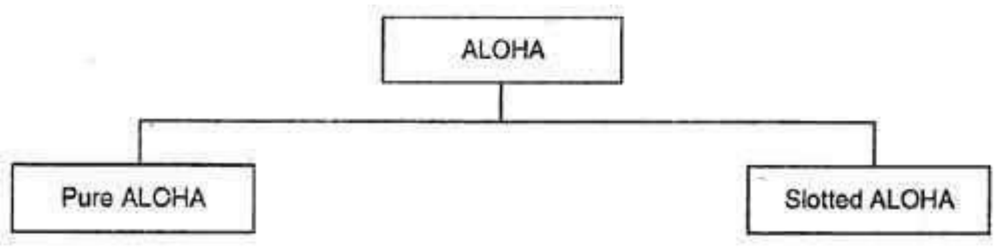
ALOHA

ALOHA: ALOHA is a system for coordinating and arbitrating access to a shared communication Networks channel. It was developed in the 1970s by Norman Abramson and his colleagues at the University of Hawaii. The original system used for ground based radio broadcasting, but the system has been implemented in satellite communication systems.

A shared communication system like ALOHA requires a method of handling collisions that occur when two or more systems attempt to transmit on the channel at the same time. In the ALOHA system, a node transmits whenever data is available to send. If another node transmits at the same time, a collision occurs, and the frames that were transmitted are lost. However, a node can listen to broadcasts on the medium, even its own, and determine whether the frames were transmitted.

Aloha means "Hello". Aloha is a multiple access protocol at the datalink layer and proposes how multiple terminals access the medium without interference or collision. In 1972 Roberts developed a protocol that would increase the capacity of aloha two fold. The Slotted Aloha protocol involves dividing the time interval into discrete slots and each slot interval corresponds to the time period of one frame. This method requires synchronization between the sending nodes to prevent collisions.

There are two different versions of ALOHA



Types of ALOHA

Pure ALOHA

• **In** pure ALOHA, the stations transmit frames whenever they have data to send.

• When two or more stations transmit simultaneously, there is collision and the frames are destroyed.

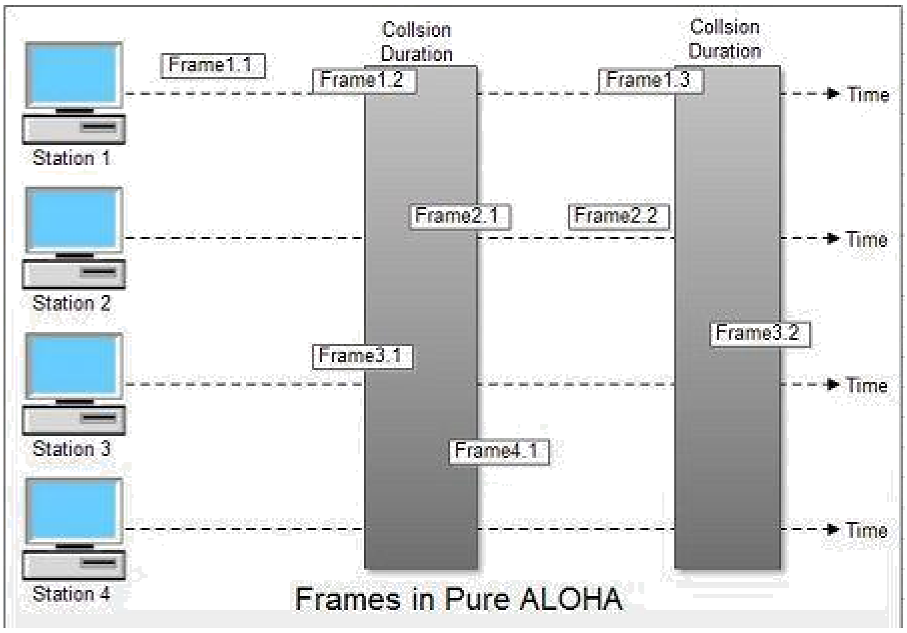
In pure ALOHA, whenever any station transmits a frame, it expects the acknowledgement from the receiver.

• If acknowledgement is not received within specified time, the station assumes that the frame (or acknowledgement) has been destroyed.

• If the frame is destroyed because of collision the 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.

• Therefore pure ALOHA dictates that when time-out period passes, each station must wait for a random amount of time before resending its frame. This randomness will help avoid more collisions.

• Figure shows an example of frame collisions in pure ALOHA.



• In fig there are four stations that .contended with one another for access to shared channel. All these stations are transmitting frames. Some of these frames collide because multiple frames are in contention for the shared channel. Only two frames, frame 1.1 and frame 2.2 survive. All other frames are destroyed.

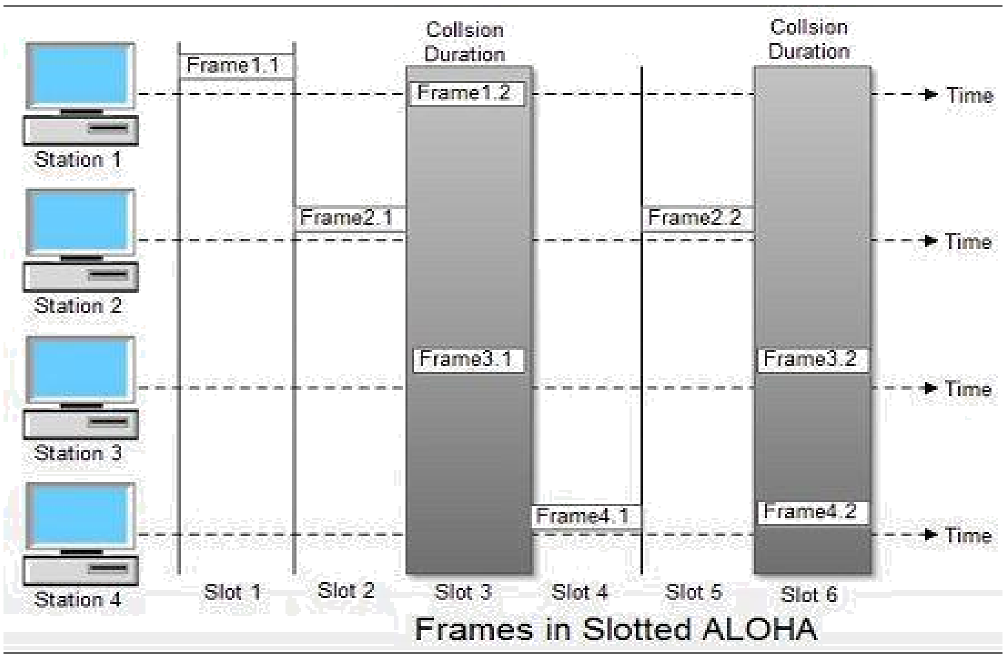
• Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be damaged. If first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted.

Slotted ALOHA

• Slotted ALOHA was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high.

• In slotted ALOHA, the time of the shared channel is divided into discrete intervals called slots.

• The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.



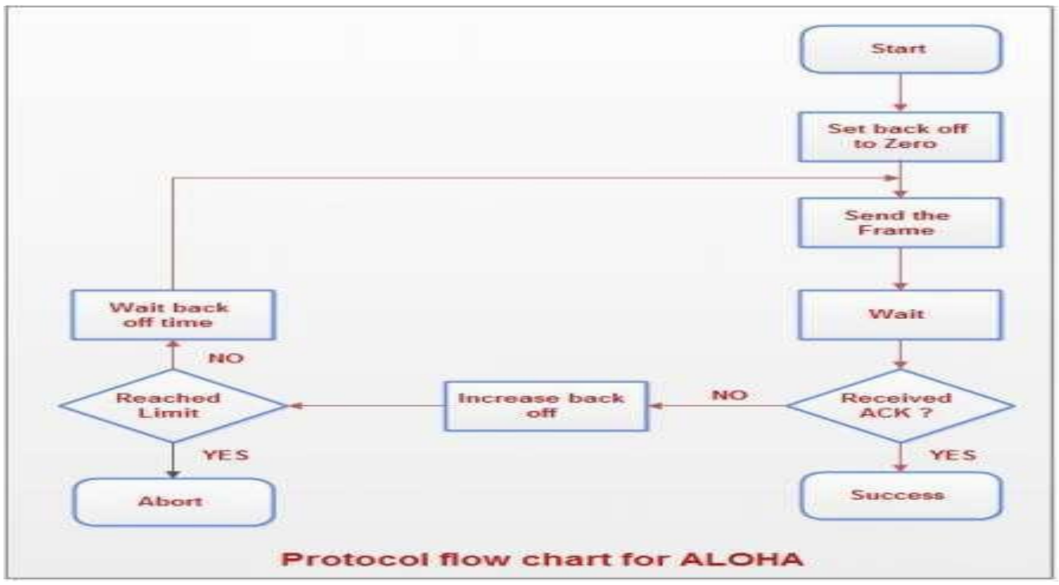
• In slotted ALOHA, if any station is not able to place the frame onto the channel at the beginning of the slot *i.e.* it misses the time slot then the station has to wait until the beginning of the next time slot.

• In slotted ALOHA, there is still a possibility of collision if two stations try to send at the beginning of the same time slot as shown in fig.

• Slotted ALOHA still has an edge over pure ALOHA as chances of collision are reduced to one-half.

Protocol Flow Chart for ALOHA

Fig. shows the protocol flow chart for ALOHA.



Explanation:

• A station which has a frame ready will send it.

• Then it waits for some time.

• If it receives the acknowledgement then the transmission is successful.

• Otherwise the station uses a backoff strategy, and sends the packet again.

• After many times if there is no acknowledgement then the station aborts the idea of transmission.

<u>CSMA(</u>Carrier Sense Multiple Access Protocols)

With slotted ALOHA the best channel utilization that can be achieved is 1/e. This is hardly surprising, since with stations transmitting at will, without paying attention to what the other stations are doing, there are bound to be many collisions. In local area networks, however, it is possible for stations to detect what other stations are doing, and adapt their behaviour accordingly. These networks can achieve a much better utilization than 1/e. In this section we will discuss some protocols for improving performance. Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called carrier sense protocols. A number of them have been proposed. Kleinrock and Tobagi (1975) have analysed several such protocols in detail. Below we will mention several versions of the carrier sense protocols.

<u>Persistent CSMA:</u>

The first carrier sense protocol that we will study here is called **1-persistent CSMA** (Carrier Sense Multiple Access). When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. If the channel is busy, the station waits until it becomes idle. When the station detects an idle channel, it transmits a frame. If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmits with a probability of 1 when it finds the channel idle.

The propagation delay has an important effect on the performance of the protocol. There is a small chance that just after a station begins sending, another station will become ready to send and sense the channel. If the first station's signal has not yet reached the second one, the latter will sense an idle channel and will also begin sending, resulting in a collision. The longer the propagation delay, the more important this effect becomes, and the worse the performance of the protocol. Even if the propagation delay is zero, there will still be collisions. If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends and then both will begin transmitting exactly simultaneously, resulting in a

collision. If they were not so impatient, there would be fewer collisions. Even so, this protocol is far better than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame. Intuitively, this approach will lead to a higher performance than pure ALOHA. Exactly the same holds for slotted ALOHA.

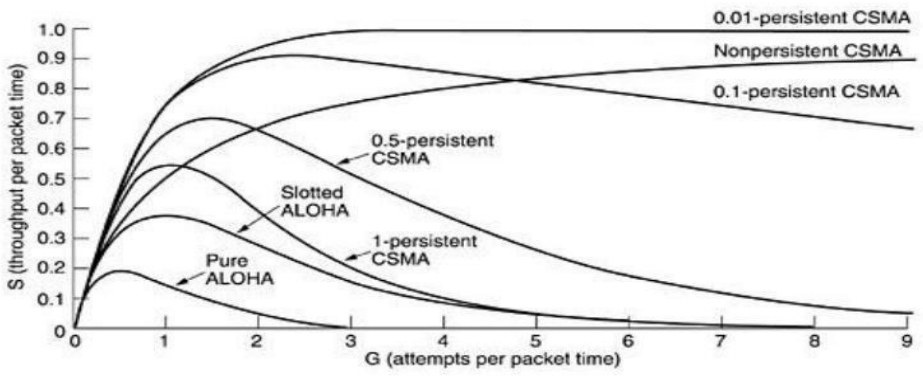
Non-persistent CSMA:

A second carrier sense protocol is **nonpersistent CSMA**. In this protocol, a conscious attempt is made to be less greedy than in the previous one. Before sending, a station senses the channel. If no one else is sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

P-persistent CSMA

The last protocol is **p-persistent CSMA**. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability q = 1 - p, it defers until the next slot. If that slot is also idle, It either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting. In the latter case, the unlucky station acts as if there had been a collision (i.e., it waits a random time and starts again). If the station initially senses the channel busy, it waits until the next slot and applies the above algorithm.

Figure 4 shows the computed throughput versus offered traffic for all three protocols, as well as for pure and slotted ALOHA.



CSMA with Collision Detection

Persistent and non persistent CSMA protocols are clearly an improvement over ALOHA because they ensure that no station begins to transmit when it senses the channel busy. Another improvement is for stations to abort their transmissions as soon as they detect a collision. In other words, if two stations sense the channel to be idle and begin transmitting simultaneously, they will both detect the collision almost immediately. Rather than finish transmitting their frames, which are irretrievably garbled anyway, they should abruptly stop transmitting as soon as the collision is detected. Quickly terminating damaged frames saves time and bandwidth.

This protocol, known as CSMA/CD (CSMA with Collision Detection) is widely used on LANs in the MAC sublayer. In particular, it is the basis of the popular Ethernet LAN, so it is worth devoting some time to looking at it in detail. CSMA/CD, as well as many other LAN protocols, uses the conceptual model of below fig. At the point marked t0, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision.

Collisions can be detected by looking at the power or pulse width of the received signal and comparing it to the transmitted signal.

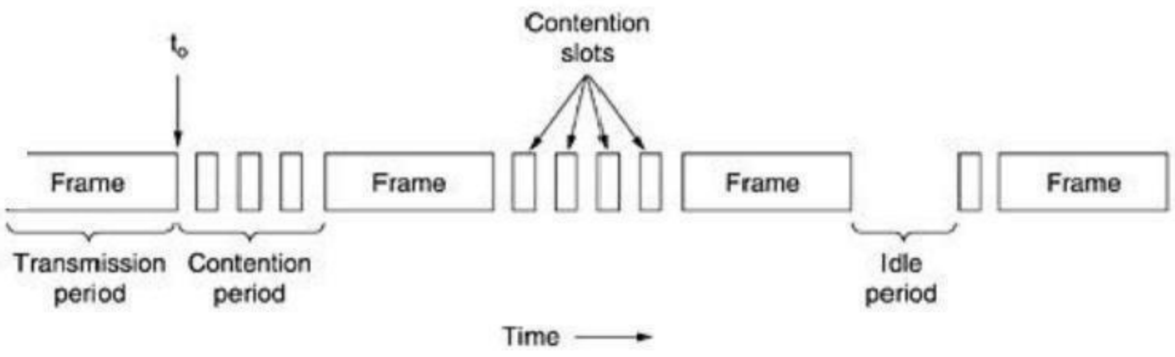


Fig. CSMA/CD can be in one of three states: contention, transmission, or idle

After a station detects a collision, it aborts its transmission, waits a random period of time, and then tries again, assuming that no other station has started transmitting in the meantime. Therefore, our model for CSMA/CD will consist of alternating contention and transmission periods, with idle periods occurring when all stations are quiet (e.g., for lack of work).

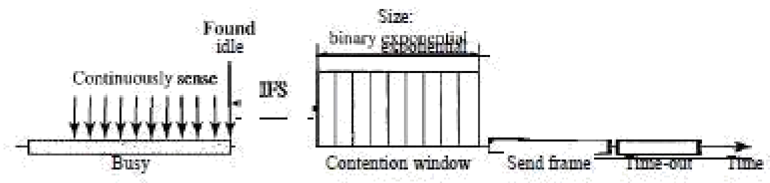
Now let us look closely at the details of the contention algorithm. Suppose that two stations both begin transmitting at exactly time t0. How long will it take them to realize that there has been a collision? The answer to this question is vital to determining the length of the contention period and hence what the delay and throughput will be. The minimum time to detect the collision is then just the time it takes the signal to propagate from one station to the other.

Based on this reasoning, you might think that a station not hearing a collision for a time equal to the full cable propagation time after starting its transmission could be sure it had seized the cable. By "seized," we mean that all other stations knew it was transmitting and would not interfere. This conclusion is wrong. Consider the following worst-case scenario. Let the time for a signal to propagate between the two farthest stations be . At t0, one station begins transmitting. At , an instant before the signal arrives at the most distant station, that station also begins transmitting. Of course, it detects the collision almost instantly and stops, but the little noise burst caused by the collision does not get back to the original station until time . In other words, in the worst case a station cannot be sure that it has seized the channel until it has transmitted for without hearing a collision. For this reason we will model the contention interval as a slotted ALOHA system with slot width . On a 1-km long coaxial cable, . For simplicity we will assume that each slot contains just 1 bit. Once the channel has been seized, a station can transmit at any rate it wants to, of course, not just at 1 bit per sec.

CSMA with Collision Avoidance:

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

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles. However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection. We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance *(CSMAlCA)* was invented for this network. Collisions are avoided through the use of CSMAICA's three strategies: the inter frame space, the contention window, and acknowledgments, as shown in below figure.



Interframe Space (IFS)

First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the inter frame space or IFS. Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time (described next). The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

Contention Window

The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back-off strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the p-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

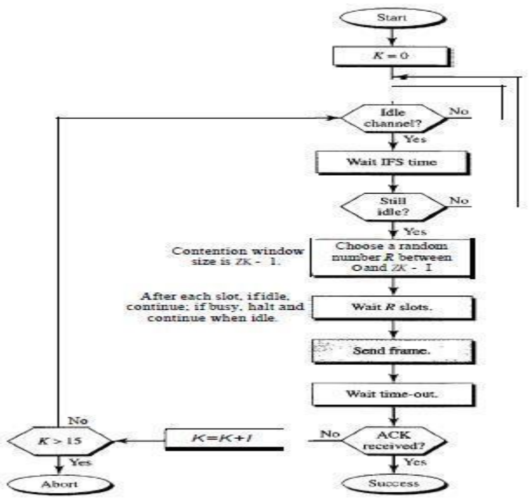
Acknowledgment

With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

Procedure

Figure 12.17 shows the procedure. Note that the channel needs to be sensed before and after the IFS. The channel also needs to be sensed during the contention time. For each time slot of the contention window, the channel is sensed. If it is found idle, the timer continues; if the channel is found busy, the timer is stopped and continues after the timer becomes idle again.

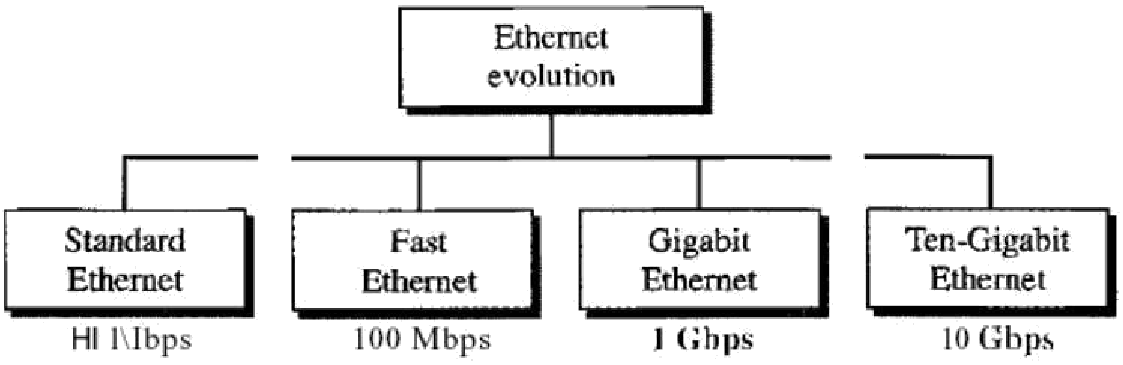
CSMAICA and Wireless Networks. CSMAICA was mostly intended for use in wireless networks. The procedure described above, however, is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals. We will see how these issues are solved by augmenting the above protocol with hand-shaking features.


Start
K = 0
Idle channel?
No
Yes
Wait IFS time
Still idle?
No
Yes
Contention window size is 2K - 1.
Choose a random number R between 0 and 2K - 1
After each slot, if idle, continue; if busy, halt and continue when idle.
Wait R slots.
Send frame.
Wait time-out.
No
K > 15
K=K+1
No
ACK received?
Yes
Abort
Yes
Success

Standard Ethernet

The Ethernet has under gone four evolutions so far as depicted in the following figure.The detailed description of different evolutions of ether has given below.

Figure 1 Ethernet evolution through four generations



MAC Sublayer

In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRC. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium.

Acknowledgments must be implemented at the higher layers. The format of the MAC frame is shown in Figure 2.

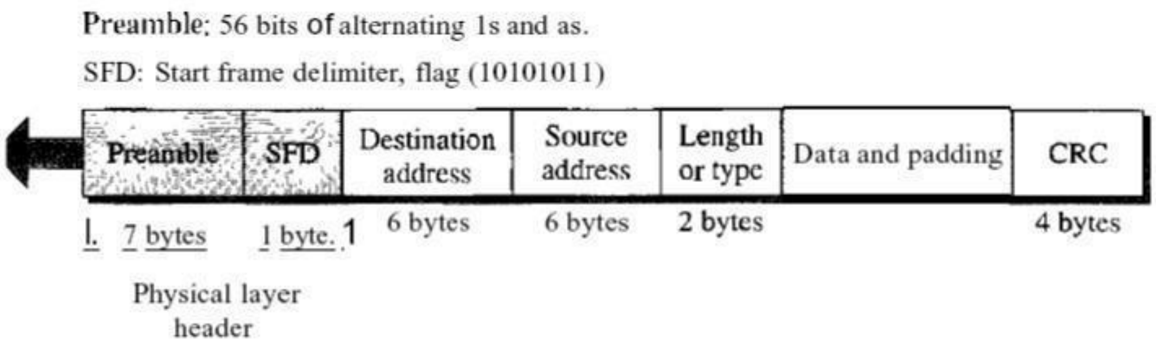


Figure 2 802.3 MAC frame

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

- Start frame delimiter (SFD). The second field (l byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.
- Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.
- Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet. We will discuss addressing shortly.

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

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

CRC: The last field contains error detection information, in this case a CRC-32

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in fig.

The minimum length restriction is required for the correct operation of CSMA/CD. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 -18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference. The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Frame length:

Minimum: 64 bytes (512 bits)

Maximum: 1518 bytes (12,144 bits)

Addressing

Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the stationwith a 6-byte 87 physical address. As shown in Fig 6, the Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.

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

6 bytes =12 hex digits =48 bits.

Unicast, Multicast,and Broadcast Addresses A source address is always a unicast address -the frame comes from only one station. The destination address, however,can be unicast, multicast, or broadcast.

Figure 5 shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.

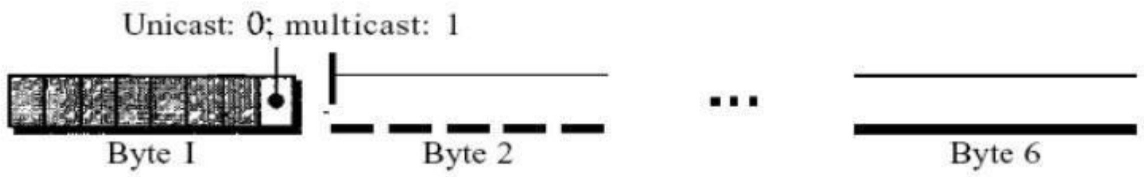


Figure 5 Unicast and multicast addresses

A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one – to - one. A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one-to-many. The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty- eight.

Physical Layer

The Standard Ethernet defines several physical layer implementations;
four of the most common, are shown in Figure 8.

Encoding and Decoding

All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the received signal is interpreted as Manchester and decoded into data. Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure 6 shows the encoding scheme for Standard Ethernet.

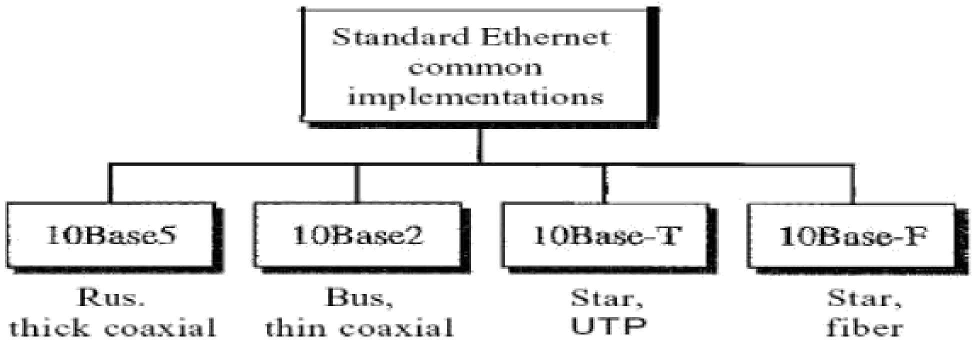


Figure 6 Categories of Standard Ethernet

Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

Types of Hub

- **Active Hub :-** These are the hubs which have their own power supply and can clean , boost and relay the signal along the network. It serves both as a repeater as well as wiring center. These are used to extend maximum distance between nodes.

- **Passive Hub :-** These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can't be used to extend distance between nodes.

Bridge – A bridge operates at data link layer. A bridge is a repeater, with add on functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

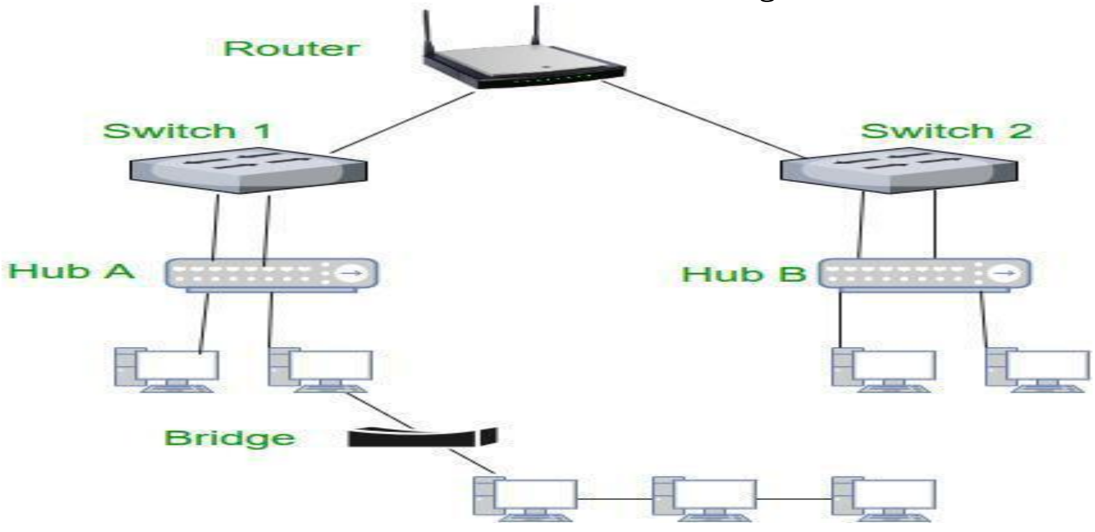
Types of Bridges

- **Transparent Bridges :-** These are the bridge in which the stations are completelyunawareofthe bridge‘s existence i.e. whether or not a bridge is added or deleted from the network,reconfiguration of the stations is unnecessary. These bridges makes use of two processes .e. bridge forwarding and bridge learning.

- **Source Routing Bridges :-** In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The hot can discover frame by sending a specical frame called discovery frame, which spreads through the entire network using all possible paths to destination.

Switch – A switch is a multi port bridge with a buffer and a design that can boost its efficiency(large number of ports imply less traffic) and performance. Switch is data link layer device. Switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

Routers – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.



Gateway – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically works as the messenger agents that take data from one system,

interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

UNIT-III

Network Layer :

The layer that responsible source to destination to delivery of packet multiple network is called network layer.

Characteristics :
i) Logical addressing
ii) Routing
iii) Connectingdifferentnetwork.

The network layer design issues

1) Store and formed packet switching.

2) Service provided to the transport layer.

3) Implementation of connectionless service.

4) Implementation of connection-oriented source.

5) Comparison of virtual circuit and datagram submits.

1) Store and formed packet switching

i) Host transmits packet to router across LAN or oval point to point link.

ii) Packet is stored on router until fully arrived and processed.

iii) Packet is forward to next router.

2) Service provide to transport layer

The network layer services have been designed with the goals : -

i) The advice should independent of router telenet

ii) The transport layer should be shilded from the number type and topology of the router present.

iii) The network addresses maid arailable to transport

4) Implementation of Connection-Less Service

- Connectionless service is offered packets are injected into the subnet individually and routed idependently of each other. Each packet is transmitte indenpendently. Connectionless service used in network layer ID and transport layer. Packet are frequently called datagram connectionless service is largly for data communication the internet.

4) Implementation of Connection-Oriented Service

Connection-oriented service is used a path from the source router to the destination router must be established before any data packet can be sent.

Connection oriented service also called virtual circuit service. This service used network layer for ATM. It also used in transport layer for TCP.

A connection must be established before any can be sent packets order preserved logical connection is also established here.

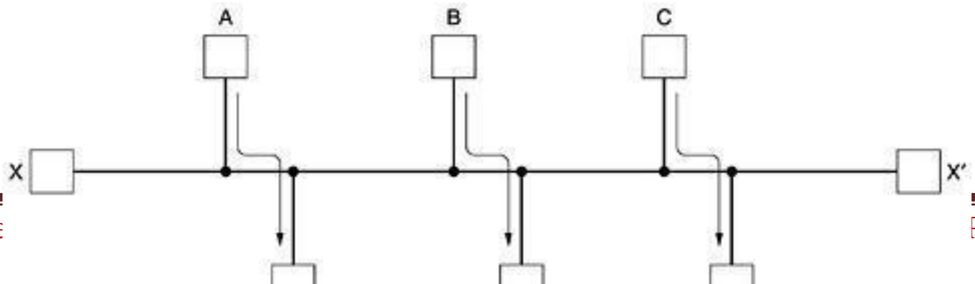
ROUTING ALGORITHMS

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.

PROPERTIES OF ROUTING ALGORITHM:

Correctness, simplicity, robustness, stability, fairness, and optimality

FAIRNESS AND OPTIMALITY.



Fairness and optimality may sound obvious, but as it turns out, they are often contradictory goals. There is enough traffic between A and A', between B and B', and between C and C' to saturate the horizontal links. To maximize the total flow, the X to X' traffic should be shut off altogether. Unfortunately, X and X' may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed.

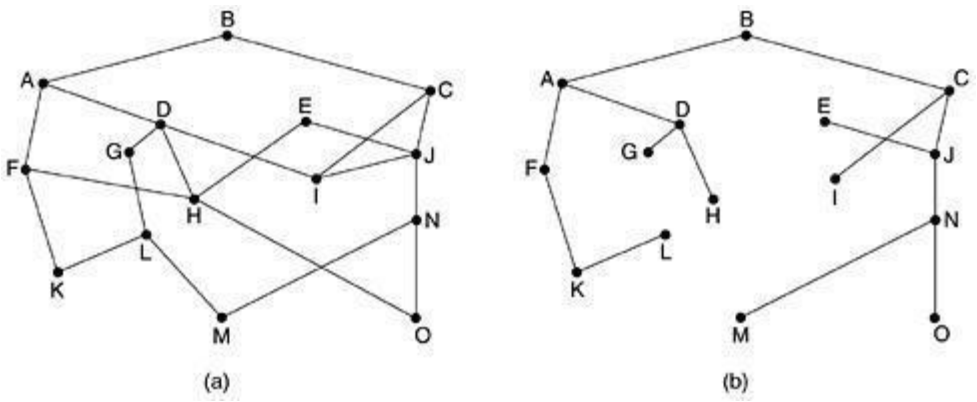
CATEGORY OF ALGORITHM

- Routing algorithms can be grouped into two major classes: **nonadaptive and adaptive. Nonadaptive algorithms** do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off-line, and downloaded to the routers when the network is booted.This procedure is sometimes called **Static routing**.
- **Adaptive algorithms**, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well.This procedure is sometimes called **dynamic routing**

THE OPTIMALITY PRINCIPLE

If router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route.

The set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree.



***(a) A subnet. (b) A sink tree for router* B.**

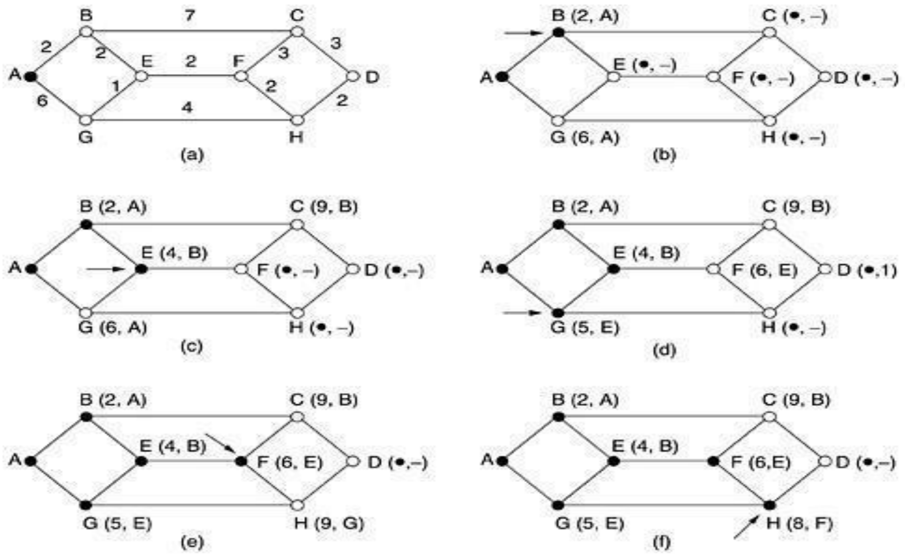
Such a tree is called a **sink tree** where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist.

The goal of all routing algorithms is to discover and use the sink trees for all routers.

SHORTEST PATH ROUTING

- A technique to study routing algorithms: The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.
- One way of measuring path length is the number of hops. Another metric is the geographic distance in kilometers. Many other metrics are also possible. For example, each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs.
- In the general case, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured

  • delay, and other factors. By changing the weighting function, the algorithm would then compute the "shortest" path measured according to
- any one of a number of criteria or to a combination of criteria.



The first five steps used in computing the shortest path from* A *to* D. *The arrows indicate the working node.

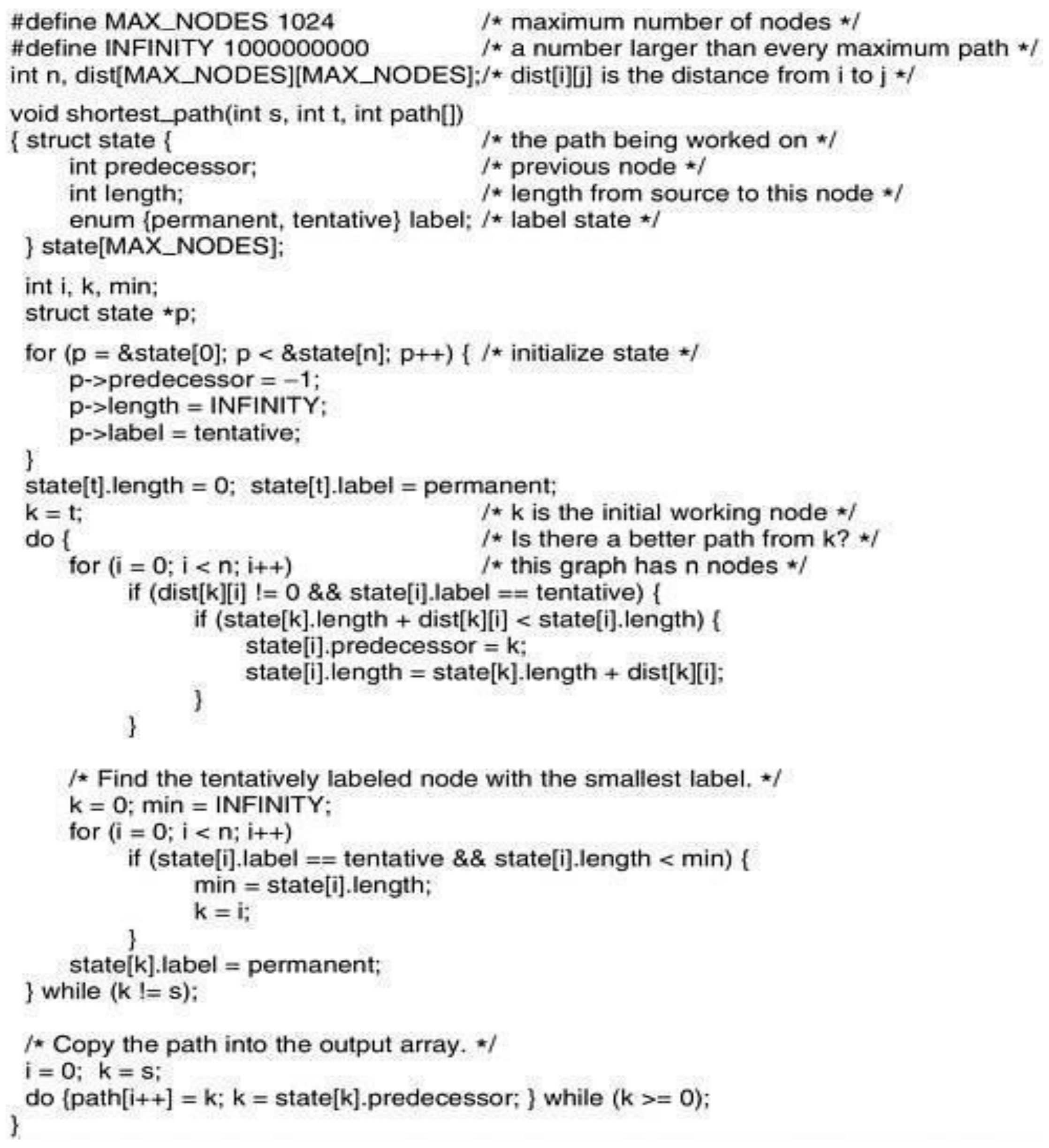
- To illustrate how the labelling algorithm works, look at the weighted, undirected graph of Fig. 5-7(a), where the weights represent, for example,distance.
- We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle.
- Then we examine, in turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to *A*.
- Whenever a node is relabelled, we also label it with the node from which the probe was made so that we can reconstruct the final path later.
- Having examined each of the nodes adjacent to *A*, we examine all the tentatively labelled nodes in the whole graph and make the one with the
- We now start at *B* and examine all nodes adjacent to it. 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, we have a shorter path, so the node is relabelled.

- After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labelled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure 5-7 shows the first five steps of the algorithm.

- then *E* has already been probed (on the round following the one when *Z* was made permanent), so the *AXYZE* path has not escaped our attention and
- thus cannot be a shorter path.
- Now consider the case where *Z* is still tentatively labelled. Either the label at *Z* is greater than or equal to that at *E*, in which case *AXYZE* cannot be a shorter path than *ABE*, or it is less than that of *E*, in which case *Z* and
- not *E* will become permanent first, allowing *E* to be probed from *Z*.
- This algorithm is given in Fig. 5-8. The global variables *n* and *dist* describe the graph and are initialized before *shortest path* is called. The only difference between the program and the algorithm described above is that in Fig. 5-8, we compute the shortest path starting at the terminal node, *t*, rather than at the source node, *s*. Since the shortest path from *t* to *s* in an

undirected graph is the same as the shortest path from *s* to *t*, it does not matter at which end we begin (unless there are several shortest paths, in which case reversing the search might discover a different one). The reason for searching backward is that each node is labelled with its predecessor rather than its successor. When the final path is copied into the output variable, *path*, the path is thus reversed. By reversing the search, the two

- effects cancel, and the answer is produced in the correct order.



```
#define MAX_NODES 1024                   /* maximum number of nodes */
#define INFINITY 1000000000              /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */
void shortest_path(int s, int t, int path[])
{ struct state {                         /* the path being worked on */
      int predecessor;                   /* previous node */
      int length;                        /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                 /* k is the initial working node */
  do {                                   /* Is there a better path from k? */
      for (i = 0; i < n; i++)            /* this graph has n nodes */
           if (dist[k][i] != 0 && state[i].label == tentative) {
                if (state[k].length + dist[k][i] < state[i].length) {
                     state[i].predecessor = k;
                     state[i].length = state[k].length + dist[k][i];
                }
           }

      /* Find the tentatively labeled node with the smallest label. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
           if (state[i].label == tentative && state[i].length < min) {
                min = state[i].length;
                k = i;
           }
      state[k].label = permanent;
  } while (k != s);

  /* Copy the path into the output array. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

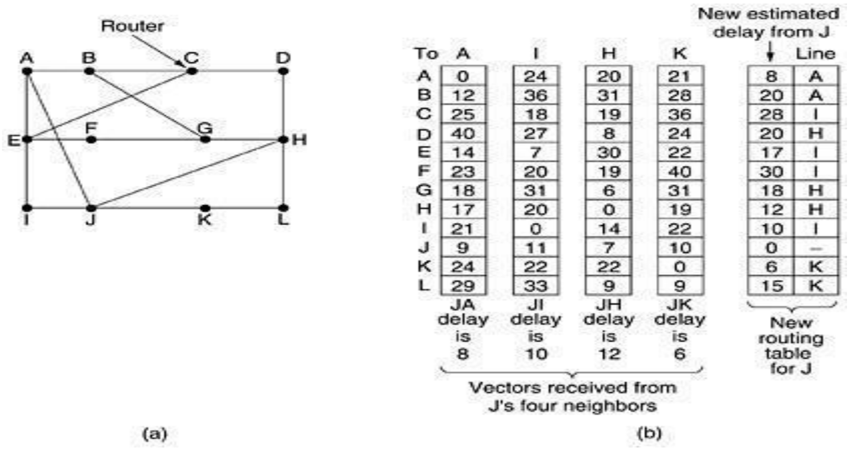
Figure 5-8. Dijkstra's algorithm to compute the shortest path through a graph.

FLOODING

- Another static algorithm is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.
- Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process.
- One such measure is to have a hop counter contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.
- Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

DISTANCE VECTOR ROUTING

- **Distance vector routing** algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to get there.
- These tables are updated by exchanging information with the neighbors.
- The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm and the **Ford-Fulkerson** algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962).
- It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

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

- Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbours 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 neighbours, *A*, *I*, *H*, and *K* as 8, 10, 12, and 6 msec, respectively.

Each node constructs a one-dimensional array containing the "distances"(costs) to all other nodes and distributes that vector to its immediate neighbors.

1. The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.
2. A link that is down is assigned an infinite cost.

Example.

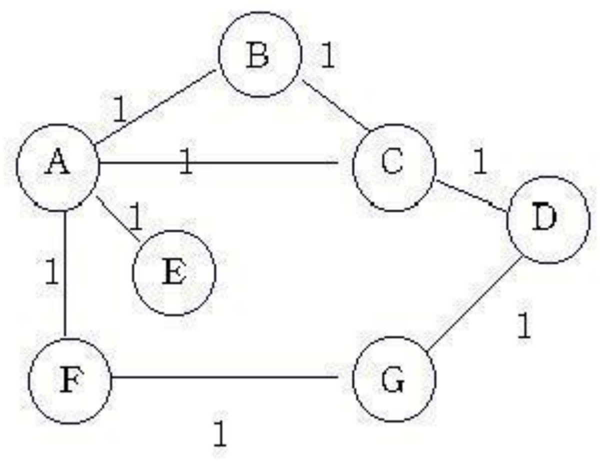


Table 1. Initial distances stored at each node(global view).

Information Stored at Node	**Distance to Reach Node**						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

We can represent each node's knowledge about the distances to all other nodes as a table like the one given in Table 1.

Note that each node only knows the information in one row of the table.

1. Every node sends a message to its directly connected neighbors containing its personal list of distance. (for example, **A** sends its information to its neighbors **B,C,E**, and **F**.)
2. If any of the recipients of the information from **A** find that **A** is advertising a path shorter than the one they currently know about, they update their list to give the new
   path length and note that they should send packets for that destination through **A**. (node **B** learns from **A** that node **E** can be reached at a cost of 1; **B** also knows it can reach **A** at a cost of 1, so it adds these to get the cost of reaching **E** by means of **A**. **B** records that it can reach **E** at a cost of 2 by going through **A**.)
3. After every node has exchanged a few updates with its directly connected neighbors, all nodes will know the least-cost path to all the other nodes.
4. In addition to updating their list of distances when they receive updates, the nodes need to keep track of which node told them about the path that they used to calculate the cost, so that they can create their forwarding table. (for example, **B** knows that it was **A** who said " I can reach **E** in one hop" and so **B** puts an entry in its table that says " To reach **E**, use the link to **A**.)

Table 2. final distances stored at each node (global view).

Information Stored at Node	**Distance to Reach Node**						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D		2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G		3	2	1	3	1	0

In practice, each node's forwarding table consists of a set of triples of the form:

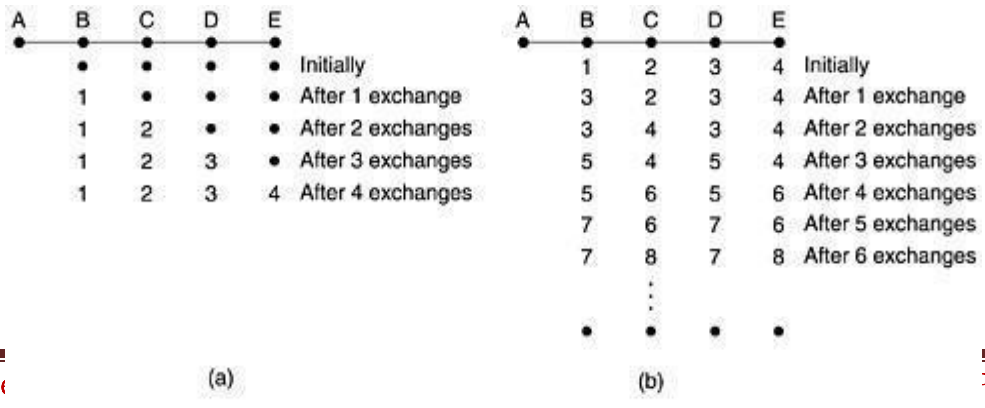
(Destination, Cost, NextHop).

For example, Table 3 shows the complete routing table maintained at node B for the network in figure1.

Table 3. Routing table maintained at node B.

Destination	Cost
A	1
C	1
D	2
E	2
F	2
G	3

THE COUNT-TO-INFINITY PROBLEM



A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

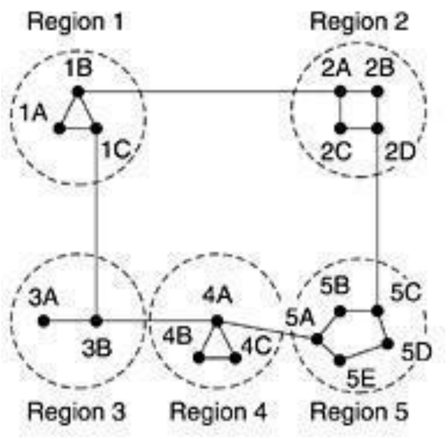
(b)

- ➢ Consider the five-node (linear) subnet of Fig.(a), where the delay metric is the number of hops. Suppose *A* is down initially and all the other routers know this. In other words, they have all recorded the delay to *A* as infinity.
- ➢ Now let us consider the situation of Fig.(b), in which all the lines and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4, respectively. Suddenly *A* goes down, or alternatively, the line

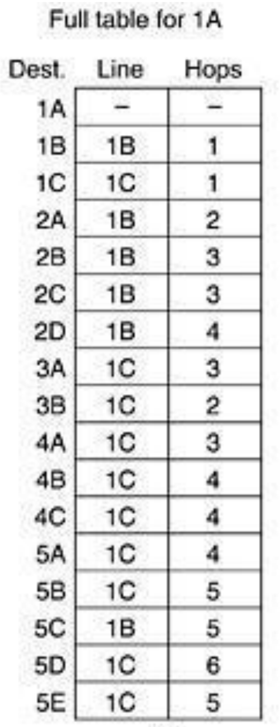
  between *A* and *B* is cut, which is effectively the same thing from *B*'s point of view.

HIERARCHICAL ROUTING

- The routers are divided into what we will call regions, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.
- For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on, until we run out of names for aggregations.



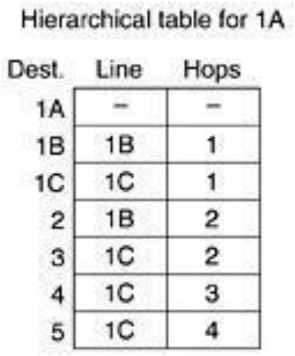
(a)



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)

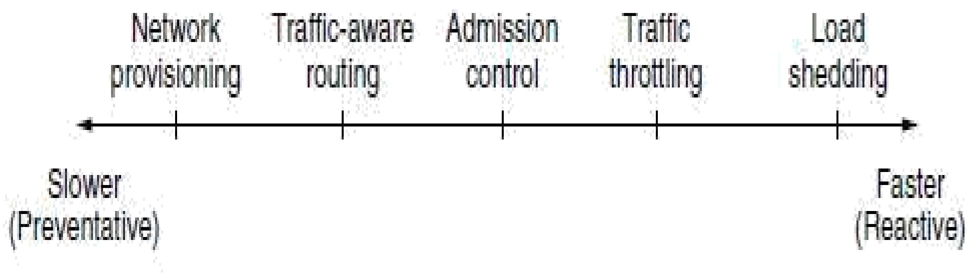


Figure 5-15 gives a quantitative example of routing in a two-level hierarchy with five regions.

- The full routing table for router 1*A* has 17 entries, as shown in Fig. 5-15(b).
- When routing is done hierarchically, as in Fig. 5-15(c), there are entries for all the local routers as before, but all other regions have been condensed into a single router, so all traffic for region 2 goes via the 1*B* - 2*A* line, but the rest of the remote traffic goes via the 1*C* -3*B* line.
- Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.

CONGESTION CONTROL ALGORITHMS

When too many packets are present in (a part of) the subnet, performance degrades. This situation is called **congestion**.

Figure 5-25 depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent.

However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered.

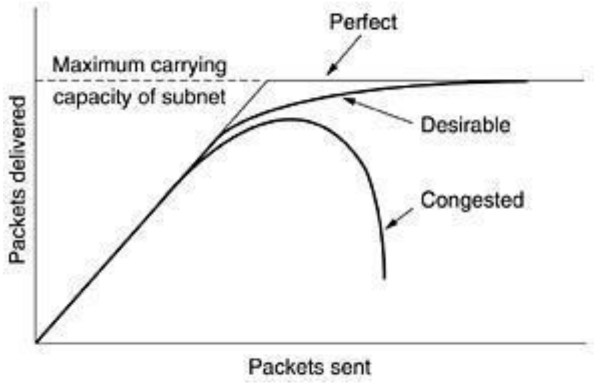


Figure 5-25. When too much traffic is offered, congestion sets in and performance degrades sharply.

- Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up.
- If there is insufficient memory to hold all of them, packets will be lost.
- Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queuing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion.

APPROACHES TO CONGESTION CONTROL

Many problems in complex systems, such as computer networks, can be viewed from a control theory point of view. This approach leads to dividing all solutions into two groups: open loop and closed loop.

- Open loop solutions attempt to solve the problem by good design.
- Tools for doing open-loop control include deciding when to accept new traffic, deciding when to discard packets and which ones, and making scheduling decisions at various points in the network.

Closed loop solutions are based on the concept of a feedback loop.

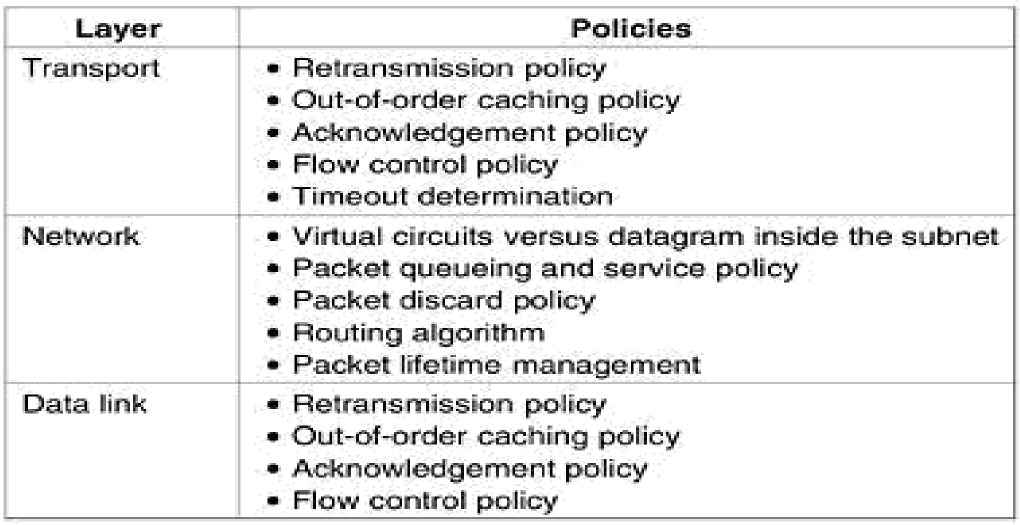
This approach has three parts when applied to congestion control:

1. Monitor the system to detect when and where congestion occurs.
2. Pass this information to places where action can be taken.
3. Adjust system operation to correct the problem.

- A variety of metrics can be used to monitor the subnet for congestion. Chief among these are the percentage of all packets discarded for lack of buffer space, the average queue lengths, the number of packets that time out and are retransmitted, the average packet delay, and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion.
- The second step in the feedback loop is to transfer the information about the congestion from the point where it is detected to the point where something can be done about it.

CONGESTION PREVENTION POLICIES

The methods to control congestion by looking at open loop systems. These systems are designed to minimize congestion in the first place, rather than letting it happen and reacting after the fact. They try to achieve their goal by using appropriate policies at various levels. In Fig. 5-26 we see different data link, network, and transport policies that can affect congestion (Jain, 1990).



Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Figure 5-26. Policies that affect congestion.

The data link layer Policies

- The **retransmission policy** is concerned with how fast a sender times out and what it transmits upon timeout. A jumpy sender that times out quickly and retransmits all outstanding packets using go back n will put a heavier load on the system than will a leisurely sender that uses selective repeat.
- Closely related to this is the **buffering policy**. If receivers routinely discard all out-of-order packets, these packets will have to be transmitted again later, creating extra load. With respect to congestion control, selective repeat is clearly better than go back n.
- **Acknowledgement policy** also affects congestion. If each packet is acknowledged immediately, the acknowledgement packets generate extra traffic. However, if acknowledgements are saved up to piggyback onto reverse traffic, extra timeouts and retransmissions may result. A tight flow control scheme (e.g., a small window) reduces the data rate and thus helps fight congestion.

The network layer Policies

- The choice between using **virtual circuits and using datagrams** affects congestion since many congestion control algorithms work only with virtual-circuit subnets.
- **Packet queueing and service policy** relates to whether routers have one queue per input line, one queue per output line, or both. It also relates to the order in which packets are processed (e.g., round robin or priority based).
- **Discard policy** is the rule telling which packet is dropped when there is no space.
- **Packet lifetime management** deals with how long a packet may live before being discarded. If it is too long, lost packets may clog up the works for a long time, but if it is too short, packets may sometimes time out before reaching their destination, thus inducing retransmissions.

The transport layer Policies

The **same issues occur as in the data link layer**, but in addition, determining the **timeout interval** is harder because the transit time across the network is less predictable than the transit time over a wire between two routers. If the timeout interval is too short, extra packets will be sent unnecessarily. If it is too long, congestion will be reduced but the response time will suffer whenever a packet is lost.

ADMISSION CONTROL

One technique that is widely used to keep congestion that has already started from getting worse is **admission control**.

Once congestion has been signaled, no more virtual circuits are set up until the problem has gone away.

An alternative approach is to allow new virtual circuits but carefully route all new virtual circuits around problem areas. For example, consider the subnet of Fig. 5-27(a), in which two routers are congested, as indicated.

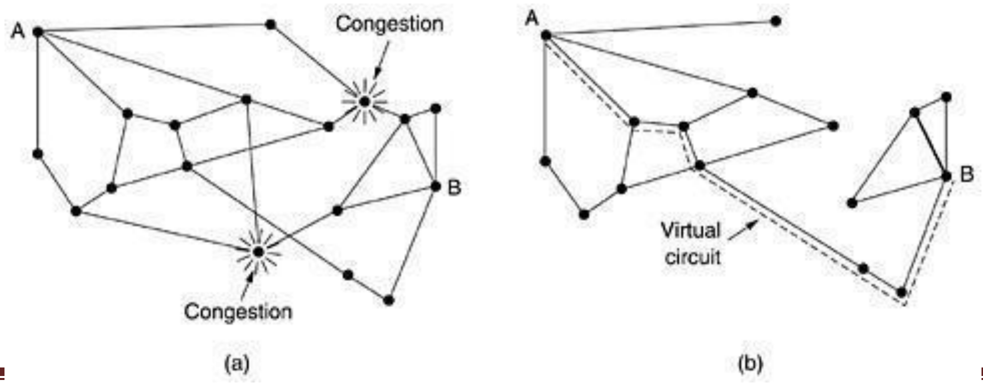


Figure 5-27. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion.
A virtual circuit from* A *to* B *is also shown.

Suppose that a host attached to router *A* wants to set up a connection to a host attached to router *B*. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the subnet as shown in Fig. 5-27(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

TUNNELING

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable.

This case is where the source and destination hosts are on the same type of network, but there is a different network in between.

As an example, think of an international bank with a TCP/IP-based Ethernet in Paris, a TCP/IP-based Ethernet in London, and a non-IP wide area network (e.g., ATM) in between, as shown in Fig. 5-47.

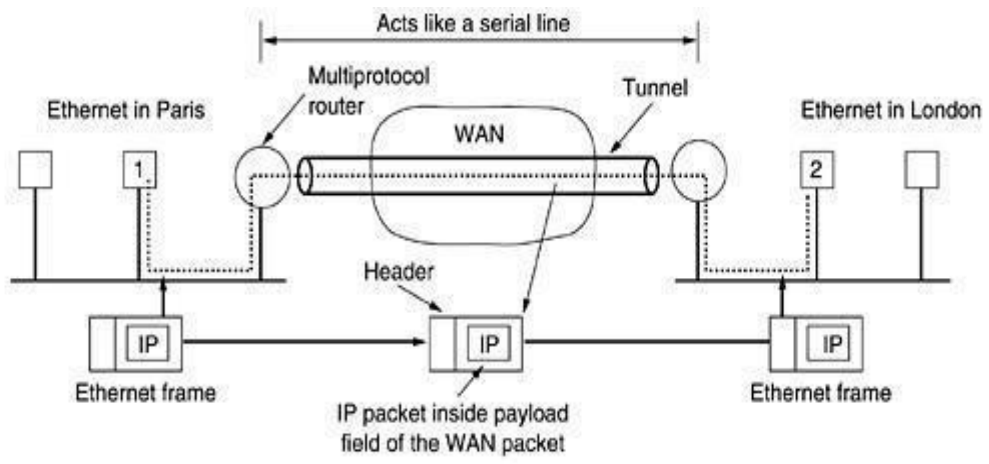


Figure 5-47. Tunneling a packet from Paris to London.

The solution to this problem is a technique called **tunneling**.

To send an IP packet to host 2, host 1 constructs the packet containing the IP address of host 2, inserts it into an Ethernet frame addressed to the Paris multiprotocol router, and puts it on the Ethernet. When the multiprotocol router gets the frame, it removes the IP packet, inserts it in the payload field of the

WAN network layer packet, and addresses the latter to the WAN address of the London multiprotocol router. When it gets there, the London router removes the IP packet and sends it to host 2 inside an Ethernet frame.

The WAN can be seen as a big tunnel extending from one multiprotocol router to the other. The IP packet just travels from one end of the tunnel to the other, snug in its nice box. Neither do the hosts on either Ethernet. Only the multiprotocol router has to understand IP and WAN packets. In effect, the entire distance from the middle of one multiprotocol router to the middle of the other acts like a serial line.

Consider a person driving her car from Paris to London. Within France, the car moves under its own power, but when it hits the English Channel, it is loaded into a high-speed train and transported to England through the Chunnel (cars are not permitted to drive through the Chunnel). Effectively, the car is being carried as freight, as depicted in Fig. 5-48. At the far end, the car is let loose on the English roads and once again continues to move under its own power. Tunneling of packets through a foreign network works the same way.

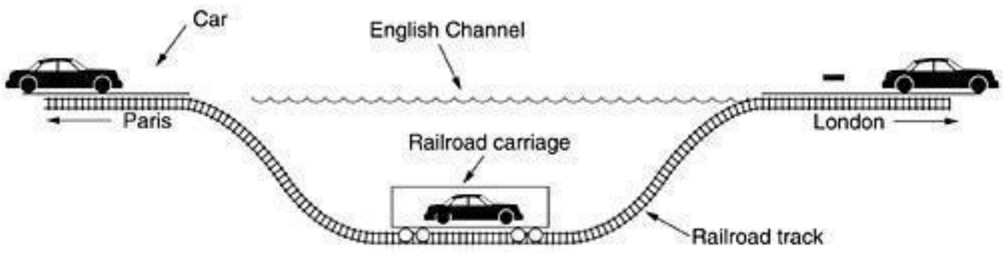


Figure 5-48. Tunneling a car from France to England.

INTERNETWORK ROUTING

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

Consider, for example, the internetwork of Fig. 5-49(a) in which five networks are connected by six (possibly multiprotocol) routers. Making a graph model of this situation is complicated by the fact that every router can directly access (i.e., send packets to) every other router connected to any network to which it is connected. For example, *B* in Fig. 5-49(a) can directly access *A* and *C* via network 2 and also *D* via network 3. This leads to the graph of Fig. 5-49(b).

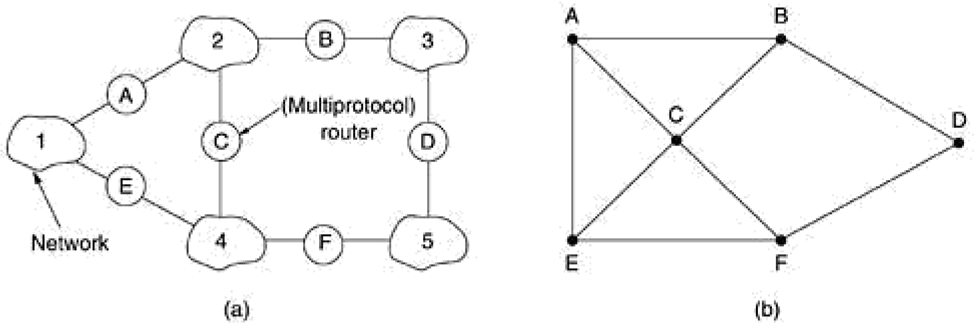


Figure 5-49. (a) An internetwork. (b) A graph of the internetwork.

Once the graph has been constructed, known routing algorithms, such as the distance vector and link state algorithms, can be applied to the set of multiprotocol routers.

This gives a two-level routing algorithm: within each network an interior gateway protocol is used, but between the networks, an exterior gateway protocol is used ("gateway" is an older term for "router").

Network in an internetwork is independent of all the others, it is often referred to as an Autonomous System (AS).

A typical internet packet starts out on its LAN addressed to the local multiprotocol router (in the MAC layer header). After it gets there, the network layer code decides which multiprotocol router to forward the packet to, using its own routing tables. If that router can be reached using the packet's native network protocol, the packet is forwarded there directly. Otherwise it is tunneled there, encapsulated in the protocol required by the intervening network. This process is repeated until the packet reaches the destination network.

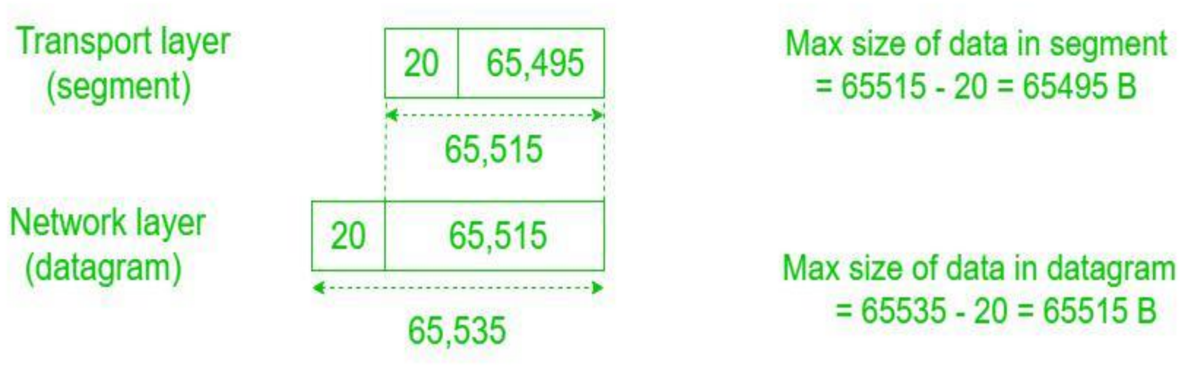
One of the differences between internetwork routing and intranet work routing is that internetwork routing may require crossing international boundaries. Various laws suddenly come into play, such as Sweden's strict privacy laws about exporting personal data about Swedish citizens from Sweden. Another example is the Canadian law saying that data traffic originating in Canada and ending in Canada may not leave the country. This law means that traffic from Windsor, Ontario to Vancouver may not be routed via nearby Detroit, even if that route is the fastest and cheapest.

Another difference between interior and exterior routing is the cost. Within a single network, a single charging algorithm normally applies.

However, different networks may be under different managements, and one route may be less expensive than another. Similarly, the quality of service offered by different networks may be different, and this may be a reason to choose one route over another.

<u>PACKET FRAGMENTATION</u>

- **Fragmentation** is done by the network layer when the maximum size of datagram is greater than maximum size of data that can be held a frame i.e., its Maximum Transmission Unit (MTU). The network layer divides the datagram received from transport layer into fragments so that data flow is not disrupted.
- Since there are 16 bits for total length in IP header so, maximum size of IP datagram = $2^{16} – 1$ = 65, 535 bytes.


Transport layer (segment)
20
65,495
65,515
Max size of data in segment = 65515 - 20 = 65495 B
Network layer (datagram)
20
65,515
65,535
Max size of data in datagram = 65535 - 20 = 65515 B

- Receiver identifies the frame with the **identification (16 bits)** field in IP header. Each fragment of a frame has same identification number.
- Receiver identifies sequence of frames using the **fragment offset(13 bits)** field in IP header
- An overhead at network layer is present due to extra header introduced due to fragmentation.

<u>Fields in IP header for fragmentation</u>

1. Identification (16 bits) – use to identify fragments of same frame.

2. Fragment offset (13 bits) – use to identify sequence of fragments in the frame. It generally indicates number of data bytes preceeding or ahead of thefragment.

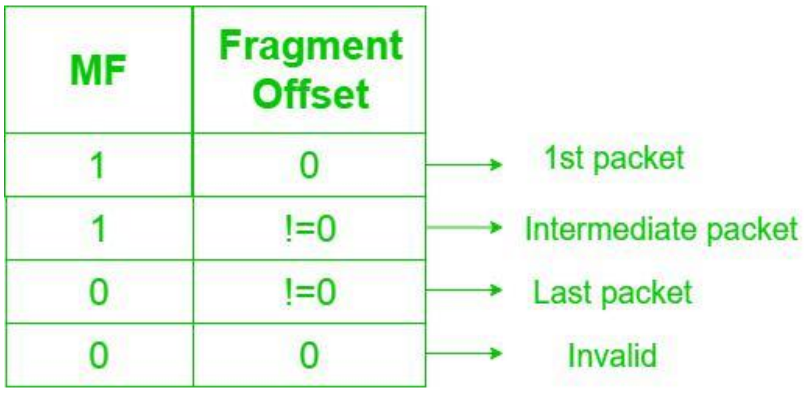
Maximum fragment offset possible = (65535 – 20) – 1 = 65514 {where 65535 is maximum size of datagram and 20 is minimum size of IP header} So, we need ceil($\log_2 65514$) = 16 bits for fragment offset but fragment offset field has only 13 bits. So, to represent efficiently we need to scale down fragment offset field by $2^{16}/2^{13}$ = 8 which acts as a scaling factor. Hence, all

More fragments (MF = 1 bit) – tells if more fragments ahead of this fragment i.e. if MF = 1, more fragments are ahead of this fragment and if MF = 0, it is the last fragment.

Don‟t fragment (DF = 1 bit) – if we don‘t want the packet to be fragmented then DF is set i.e. DF = 1.

Reassembly of Fragments –

It takes place only at destination and not at routers since packets take independent path(datagram packet switching), so all may not meet at a router and hence a need of fragmentation may arise again. The fragments may arrive out of order also.



MF	Fragment Offset	
1	0	1st packet
1	!=0	Intermediate packet
0	!=0	Last packet
0	0	Invalid

Algorithm –

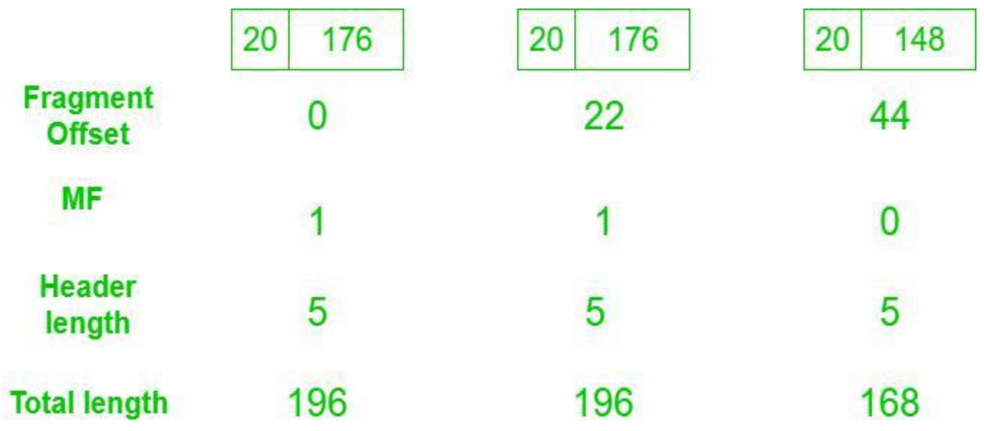
1. Destination should identify that datagram is fragmented from MF, Fragment offset field.
2. Destination should identify all fragments belonging to same datagram from Identification field.
3. Identify the 1st fragment(offset = 0).
4. Identify subsequent fragment using header length, fragment offset.
5. Repeat until MF = 0.

Efficiency –

Efficiency (e) = useful/total = (Data without header)/(Data with header)
Throughput = e * B { where B is bottleneck bandwidth }

Example – An IP router with a Maximum Transmission Unit (MTU) of 200 bytes has received an IP packet of size 520 bytes with an IP header of length 20 bytes. The values of the relevant fields in the IP header.

Explanation – Since MTU is 200 bytes and 20 bytes is header size so, maximum length of data = 180 bytes but it can be represented in fragment offset since not divisible by 8 so, maximum length of data feasible = 176 bytes.
Number of fragments = (520/200) = 3.
Header length = 5 (since scaling factor is 4 therefore, 20/4 = 5) Efficiency, e = (Data without header)/(Data with header) = 500/560 = 89.2 %



	20 | 176	20 | 176	20 | 148
Fragment Offset	0	22	44
MF	1	1	0
Header length	5	5	5
Total length	196	196	168

IPv4 and IPv6 Protocols

Definition of IPv4

An IPv4 address is a 32- bit binary value, which can be displayed as four decimal digits. The IPv4 address space offers about 4.3 billion addresses. Only 3.7 billion addresses can only be assigned out of 4.3 billion address. The other addresses are conserved for specific purposes such as multicasting, private address space, loopback testing, and research.

IP version 4 (IPv4) uses Broadcasting for transferring packets from one computer to all computers; this probably generates problems sometimes.

Dotted-Decimal Notation of IPv4 128.11.3.31

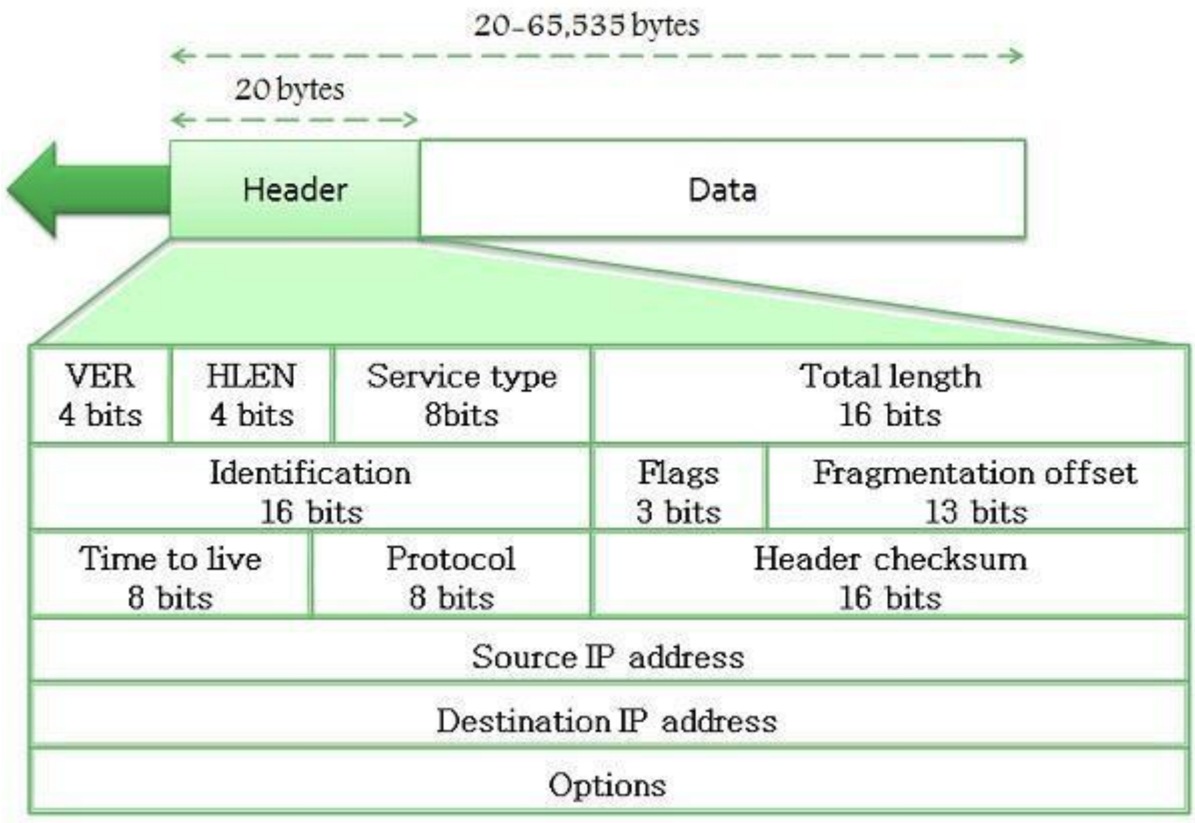
Packet Format

An IPv4 datagram is a variable-length packet comprised of a header (20 bytes) and data (up to 65,536 along with header). The header contains information essential to routing and delivery.

Base Header

Version: It defines the version number of IP, i.e., in this case, it is 4 with a binary value of 0100.
Header length (HLEN): It represents the length of the header in multiple of four bytes.

Service type: It determines how datagram should be handled and includes individual bits such as level of throughput, reliability, and delay.
Total length: It signifies the entire length of the IP datagram.
Identification: This field is used in fragmentation. A datagram is divided when it passes through different networks to match the network frame size. At that time each fragment is determined with a sequence number in this field.
Flags: The bits in the flags field handles fragmentation and identifies the first, middle or last fragment, etc.



IPv4 Datagram

Fragmentation offset: It‘s a pointer that represents the offset of the data in the original datagram.
Time to live: It defines the number of hops a datagram can travel before it is rejected. In simple words, it specifies the duration for which a datagram remains on the internet.
Protocol: The protocol field specifies which upper layer protocol data are encapsulated in the datagram (TCP, UDP, ICMP, etc.).
Header checksum: This is a 16-bit field confirm the integrity of the header values, not the rest of the packet.
Source address: It‘s a four-byte internet address which identifies the source of the datagram.
Destination address: This is a 4-byte field which identifies the final destination.
Options: This provides more functionality to the IP datagram. Furthermore can carry fields like control routing, timing, management, and alignment.

IPv4 is a two-level address structure (net id and host id) classified into five categories (A, B, C, D, and E).

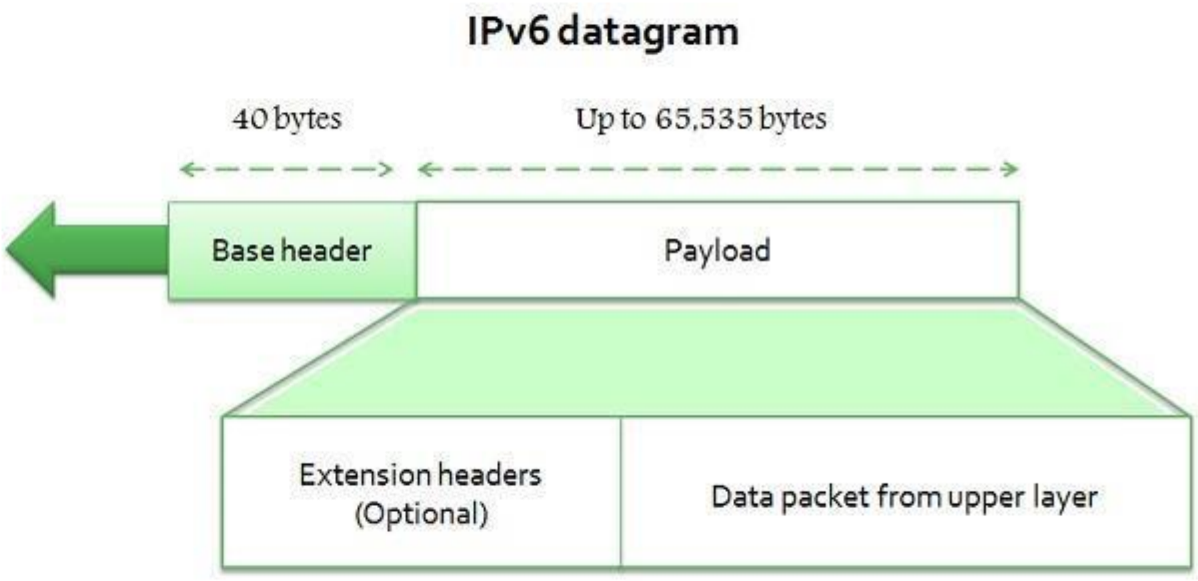
Definition of IPv6

An IPv6 address is a 128-bit binary value, which can be displayed as 32 hexadecimal digits. Colons isolate entries in a sequence of 16-bit Hexadecimal fields. It provides 3.4 x 10^{38} IP addresses. This version of IP addressing is designed to fulfill the needs of exhausting IP‘s and providing sufficient addresses for future Internet growth requirements. As IPv4 uses two-level address structure where the use of address space is insufficient. That was the reason for proposing the IPv6, to overcome the deficiencies IPv4. The format and the length of the IP addresses were changed along with the packet format and protocols were also modified.

Hexadecimal Colon Notation of IPv6 FDEC:BA98:7654:3210:ADBF:BBFF:2922:FFFF

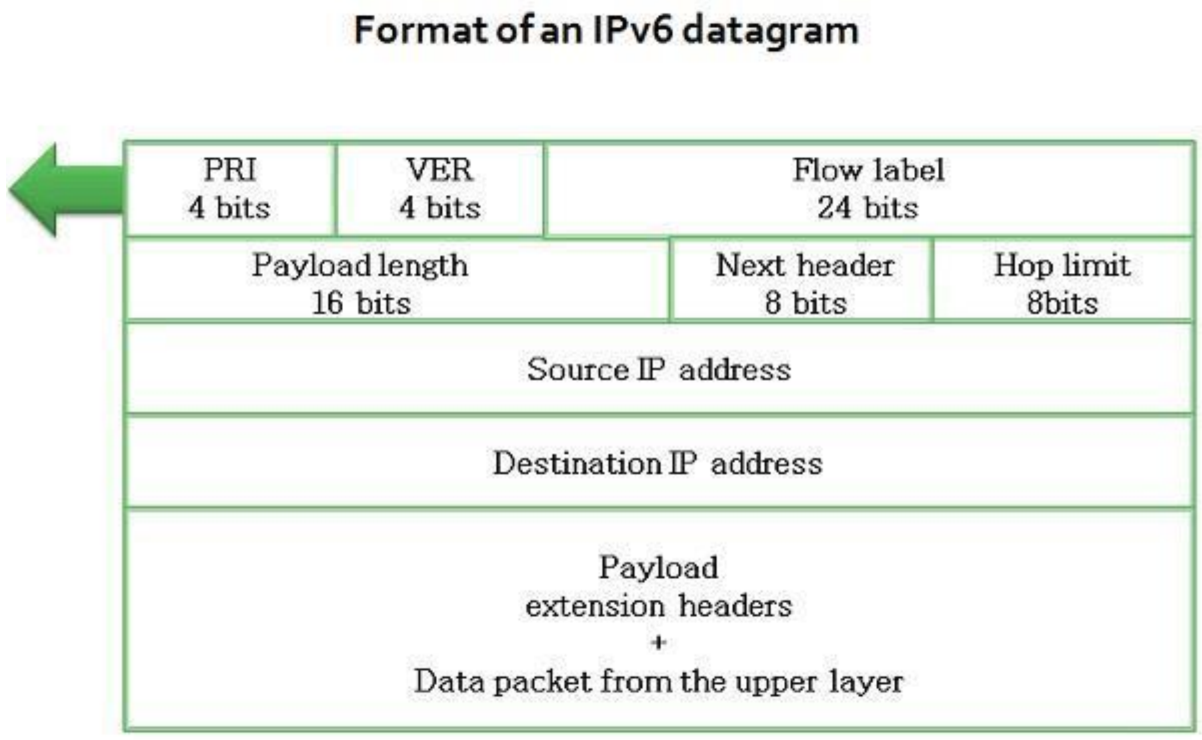
IPv6 Packet format

Each packet is consist of a mandatory base header succeeded by the payload. The payload includes two parts namely optional extension headers and data from an upper layer. The base header consumes 40 bytes, inversely the extension headers and data from the top layer usually hold up to 65,535 bytes of information.



Base Header

Version: This four-bit field specifies the version of the IP, i.e., 6 in this case. **Priority:** It defines the priority of the packet concerning traffic congestion. **Flow label:** The reason for designing this protocol is to facilitate with special controlling for a certain flow of data. **Payload length:** It defines the total length of the IP datagram excepting the base header.



Next header: It‘s an eight-bit field describe the header that trails the base header in the datagram. The next header is one of the optional extension headers which IP uses or the header for an upper layer protocol such as UDP or TCP. **Hop limit:** This eight-bit hop limit field assist with the same functions at the TTL field in IPv4.
Source address: It is a 16 bytes internet address identifies the source of the datagram.
Destination address: This is 16-byte internet address that generally describes the final destination of the datagram.

Key Differences Between IPv4 and IPv6

Let us look at the substantial difference between IPv4 and IPv6.

1. IPv4 has 32-bit address length whereas IPv6 has 128-bit address length.
2. IPv4 addresses represent the binary numbers in decimals. On the other

hand, IPv6 addresses express binary numbers in hexadecimal.

3. IPv6 uses end-to-end fragmentation while IPv4 requires an intermediate router to fragment any datagram that is too large.
4. Header length of IPv4 is 20 bytes. In contrast, header length of IPv6 is 40 bytes.
5. IPv4 uses checksum field in the header format for handling error checking. On the contrary, IPv6 removes the header checksum field.
6. In IPv4, the base header does not contain a field for header length, and 16-bit payload length field replaces it in the IPv6 header.
7. The option fields in IPv4 are employed as extension headers in IPv6.
8. The Time to live field in IPv4 refers to as Hop limit in IPv6.
9. The header length field which is present in IPv4 is eliminated in IPv6 because the length of the header is fixed in this version.
10. IPv4 uses broadcasting to transmit the packets to the destination computers while IPv6 uses multicasting and anycasting.
11. IPv6 provides authentication and encryption, but IPv4 doesn‘t provide it.

IP Addresses

An IP address, short for Internet Protocol address, is an identifying number for a piece of network hardware. Having an IP address allows a device to communicate with other devices over an IP-based network like the internet.

Most IP addresses look like this:

```
151.101.65.121
```

Other IP addresses you might come across could look more like this:

```
2001:4860:4860::8844
```

There's a lot more on what those differences mean in the *IP Versions (IPv4 vs IPv6)*section below.

What Is an IP Address Used For?

An IP address provides an identity to a networked device. Similar to a home or business address supplying that specific physical location with an identifiable address, devices on a network are differentiated from one another through IP addresses.

If I'm going to send a package to my friend in another country, I have to know the exact destination. It's not enough to just put a package with his name on it through the mail and expect it to reach him. I must instead attach a *specific* address to it, which you could do by looking it up in a phone book.

This same general process is used when sending data over the internet. However, instead of using a phone book to look up someone's name to find their physical address, your computer uses DNS servers to look up a hostname to find its IP address.

For example, when I enter a website like *www.lifewire.com* into my browser, my request to load that page is sent to DNS servers that look up that hostname (lifewire.com) to find its corresponding IP address (151.101.65.121). Without the IP address attached, my computer will have no clue what it is that I'm after.

Different Types of IP Addresses

Even if you've heard of *IP addresses* before, you may not realize that there are specific *types* of IP addresses. While all IP addresses are made up of numbers or letters, not all addresses are used for the same purpose.

There are private IP addresses, public IP addresses, static IP addresses, and dynamic IP addresses. That's quite a variety! Following those links will give you much more information on what they each mean. To add to the complexity, each type of IP address can be an IPv4 address or an IPv6 address
— again, more on these at the bottom of this page.

In short, private IP addresses are used "inside" a network, like the one you probably run at home. These types of IP addresses are used to provide a way for your devices to communicate with your router and all the other devices in your private network. Private IP addresses can be set manually or assigned automatically by your router.

Public IP addresses are used on the *outside* of your network and are assigned by your ISP. It's the main address that your home or business network uses to communicate with the rest of the networked devices around the world (i.e. the internet). It provides a way for the devices in your home, for example, to reach your ISP, and therefore the outside world, allowing them to do things like access websites and communicate directly with other people's computers.

Both private IP addresses and public IP addresses are either dynamic or static, which means that, respectively, they either change or they don't.

An IP address that is assigned by a DHCP server is a dynamic IP address. If a device does not have DHCP enabled or does not support it then the IP address must be assigned manually, in which case the IP address is called a static IP address.

How to Find Your IP Address

Different devices and operating systems require unique steps to find the IP address. There are also different steps to take if you're looking for the *public* IP address provided to you by your ISP, or if you need to see the *private* IP address that your router handed out.

Public IP Address

There are lots of ways to find your router's public IP address but sites like IP Chicken, WhatsMyIP.org, or WhatIsMyIPAddress.com make this super easy. These sites work on any network-connected device that supports a web browser, like your smartphone, iPod, laptop, desktop, tablet, etc.

Finding the private IP address of the specific device you're on isn't as simple.

Private IP Address

In Windows, you can find your device's IP address via the Command Prompt, using the **ipconfig** command.

Linux users can launch a terminal window and enter the command **hostname -I**(that's a capital "i"), **ifconfig**, or **ip addr show**.

For macOS, use the command **ifconfig** to find your local IP address. iPhone, iPad, and iPod touch devices show their private IP address through the **Settings** app in the **Wi-Fi** menu. To see it, just tap the small "i" button next to the network it's connected to.

We can see the local IP address of an Android device through **Settings** > **Wi-Fi**, or through **Settings** > **Wireless Controls** > **Wi-Fi settings** in some Android versions. Just tap on the network you're on to see a new window that shows network information that includes the private IP address.

ARP / RARP

Address Resolution Protocol / Reverse Address Resolution Protocol - to initialize the use of Internet addressing on an Ethernet or other network that uses its own media access control (MAC).

ARP allows a host to communicate with other hosts when only the Internet address of its neighbors is known. Before using IP, the host sends a broadcast ARP request containing the Internet address of the desired destination system.

16		32
Hardware Type		Protocol Type
HLen (8)	Plen (8)	Operation
Sender Hardware Address		
Sender Protocol Address		
Target Hardware Address		
Target Protocol Address		

Structure of the ARP / RARP header in 32 bit lines.

DHCP

Dynamic Host Configuration Protocol - provides Internet hosts with configuration parameters. DHCP is an extension of BOOTP. DHCP consists of two components: a protocol for delivering host-specific configuration parameters from a DHCP server to a host and a mechanism for allocation of network addresses to hosts.

8	16	24	32
Op	Htype	Hlen	Hops
Xid (4 Bytes)			
Secs (2 Bytes)		Flags (2 Bytes)	
Ciaddr (4 Bytes)			
Yiaddr (4 Bytes)			
Siaddr (4 Bytes)			
Giaddr (4 Bytes)			
Chaddr : :		(16	Bytes)

Structure of the DHCP header in 32 bit lines.

ICMP

Internet Control Message Protocol - messages which generally contain information about routing difficulties with IP datagrams or simple exchanges such as time-stamp or echo transactions.

8	16	32
Type	Code	Checksum
Identifier		Sequence number
Address mask		

Structure of the ICMP header in 32 bit lines.

CIDR

A system called Classless Inter-Domain Routing, or CIDR, was developed as an alternative to traditional subnetting. The idea is that you can add a specification in the IP address itself as to the number of significant bits that make up the routing or networking portion.

For example, we could express the idea that the IP address 192.168.0.15 is associated with the netmask 255.255.255.0 by using the CIDR notation of 192.168.0.15/24. This means that the first 24 bits of the IP address given are considered significant for the network routing.

This allows us some interesting possibilities. We can use these to reference "supernets". In this case, we mean a more inclusive address range that is not possible with a traditional subnet mask. For instance, in a class C network, like above, we could not combine the addresses from the networks 192.168.0.0 and 192.168.1.0 because the netmask for class C addresses is 255.255.255.0.

However, using CIDR notation, we can combine these blocks by referencing this chunk as 192.168.0.0/23. This specifies that there are 23 bits used for the network portion that we are referring to.

So the first network (192.168.0.0) could be represented like this in binary:

1100 0000 - 1010 1000 - 0000 0000 - 0000 0000

While the second network (192.168.1.0) would be like this:

1100 0000 - 1010 1000 - 0000 0001 - 0000 0000

The CIDR address we specified indicates that the first 23 bits are used for the network block we are referencing. This is equivalent to a netmask of 255.255.254.0, or:

1111 1111 - 1111 1111 - 1111 1110 - 0000 0000

As you can see, with this block the 24th bit can be either 0 or 1 and it will still match, because the network block only cares about the first 23 digits.

Basically, CIDR allows us more control over addressing continuous blocks of IP addresses. This is much more useful than the subnetting we talked about originally.

TRANSPORT LAYER

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.

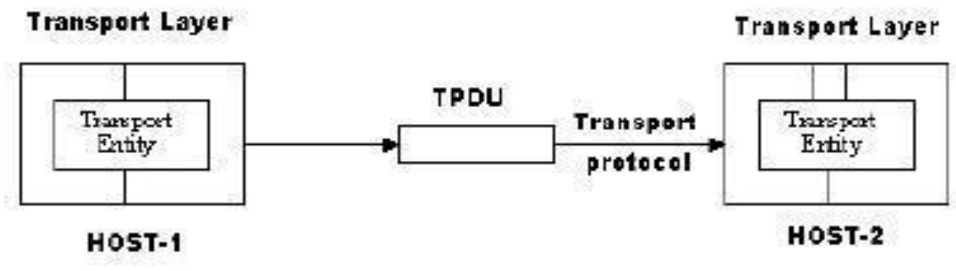
Transport Service User: The upper layers i.e., layers 5 to 7 are called Transport Service User.

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.

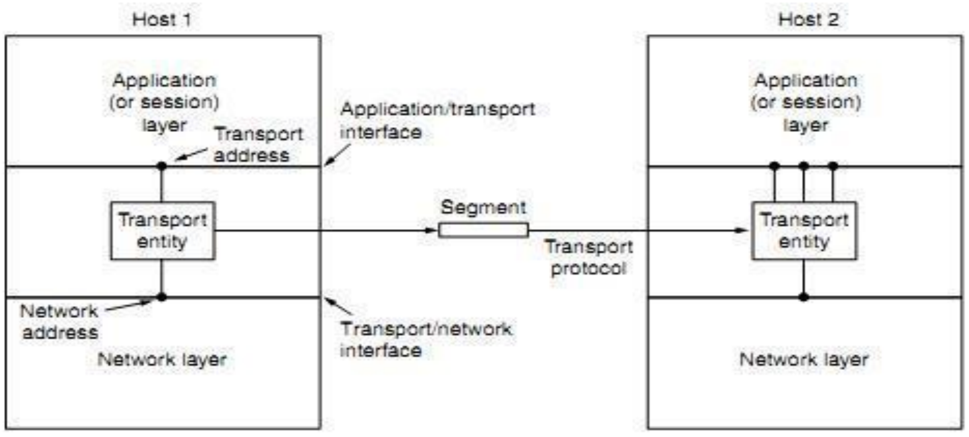


Fig 4.1: The network, Application and transport layer

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.

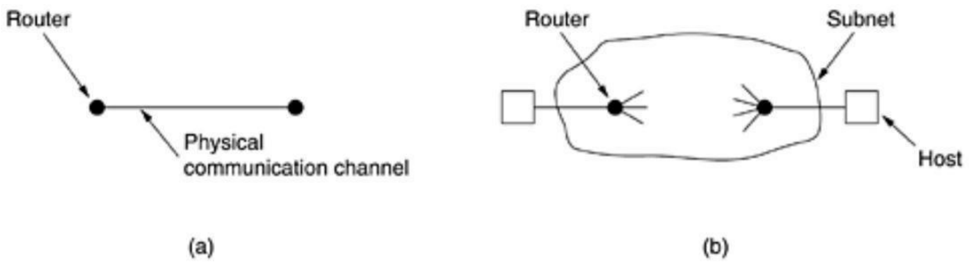


Figure (a) Environment of the data link layer. (b) Environment of the transport layer.

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.

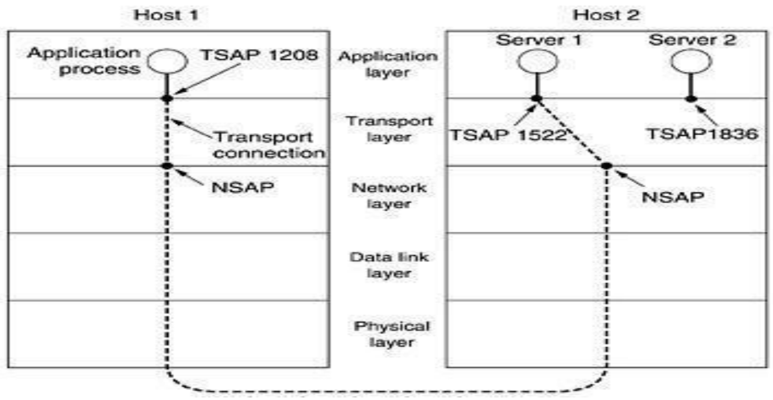


Figure 4.5 illustrates the relationship between the NSAP, TSAP and transport connection. Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP.

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. <u>ADDRESSING</u>

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.

1.<u>TSAP (Transport Service Access Point</u>) 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.

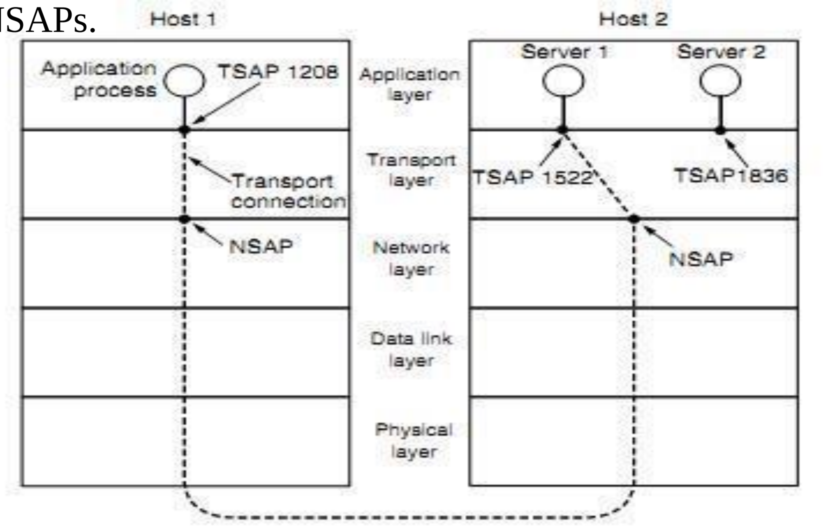


Fig 4.5: TSAP and NSAP network connections

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.

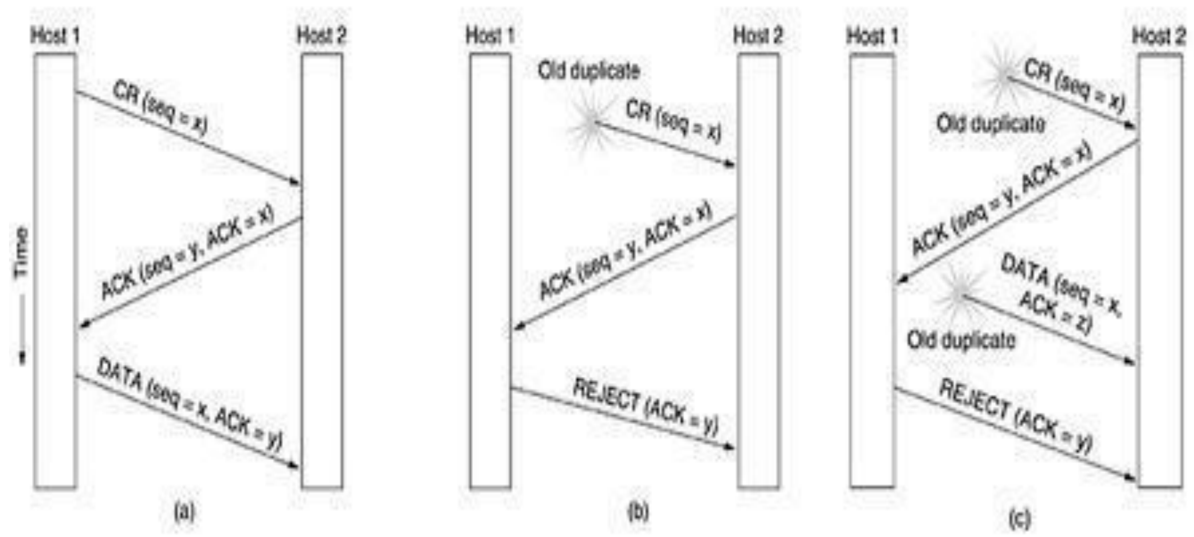


Fig 4.6: Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNEC TION REQUEST (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK

- 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:

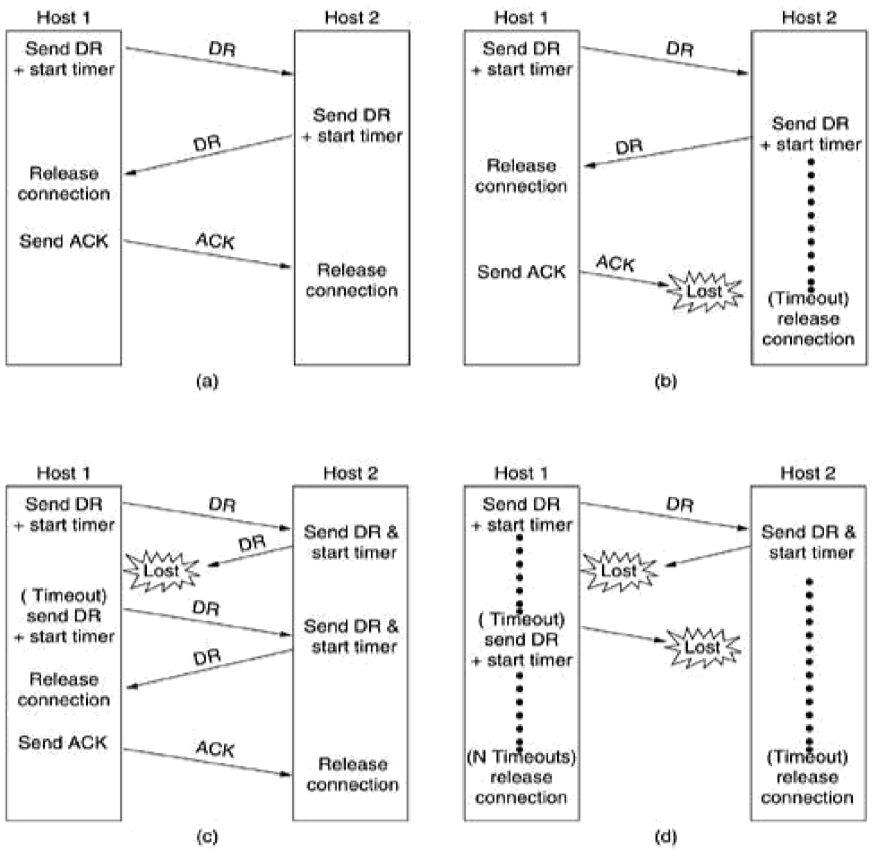
A connection is released using either asymmetric or symmetric variant. But, the improved protocol for releasing a connection is a 3-way handshake protocol.
There are two styles of terminating a connection:

Asymmetric release and
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 DRarrives, 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 expectedresponse, 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.


Host 1
Host 2
Send DR + start timer
DR
Send DR + start timer
DR
Release connection
Send ACK
ACK
Release connection
(a)
Host 1
Host 2
Send DR + start timer
DR
Send DR + start timer
DR
Release connection
Send ACK
ACK
Lost
(Timeout) release connection
(b)
Host 1
Host 2
Send DR + start timer
DR
Send DR & start timer
DR
Lost
(Timeout) send DR + start timer
DR
Send DR & start timer
DR
Release connection
Send ACK
ACK
Release connection
(c)
Host 1
Host 2
Send DR + start timer
DR
Send DR & start timer
Lost
(Timeout) send DR + start timer
Lost
(N Timeouts) release connection
(Timeout) release connection
(d)

UNIT-V

INTERNET TRANSPORT 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.

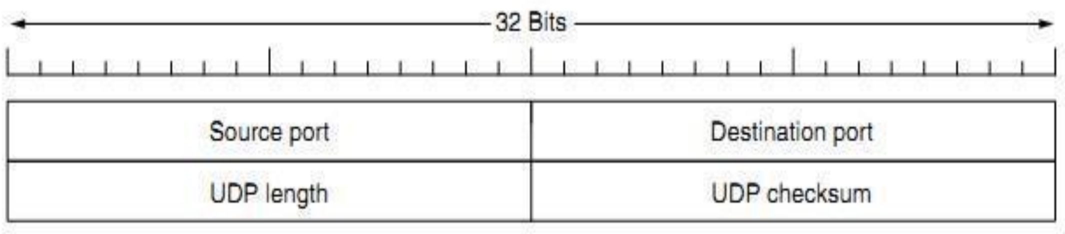


Fig 4.9: The UDP header

- **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.

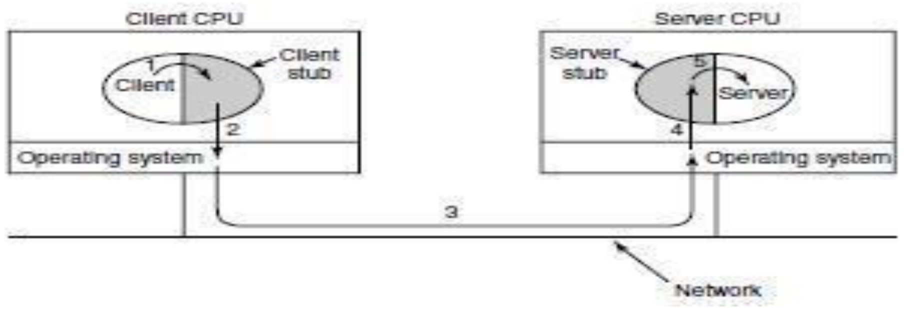


Fig 4.10: Steps in making a RPC

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:

i. The TCP Service Model
ii. The TCP Protocol
iii. The TCP Segment Header
iv. The Connection Management
v. TCP Transmission Policy
vi. TCP Congestion Control
vii. TCP Timer Management.

The TCP Service Model

TCP service is obtained by having both the sender and receiver create end points called **SOCKETS.**

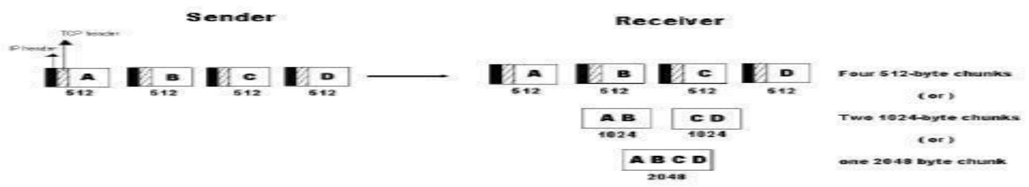
Each socket has a socket number(address)consisting of the IP address of the host, called a **“PORT”** (= TSAP)

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.

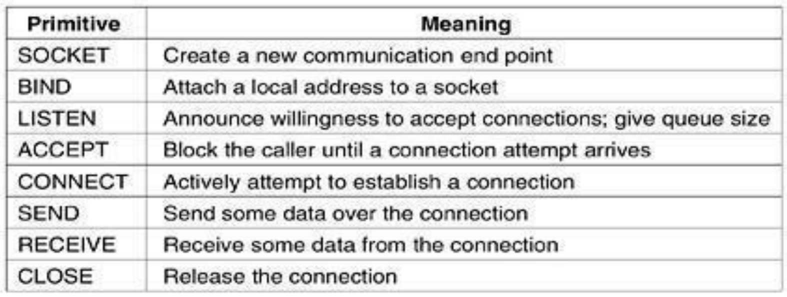
A TCP connection is a byte stream, not a message stream i.e., the data is delivered as chunks

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
PORT-23	To establish a remote login session using TELNET

The TCP Protocol

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.

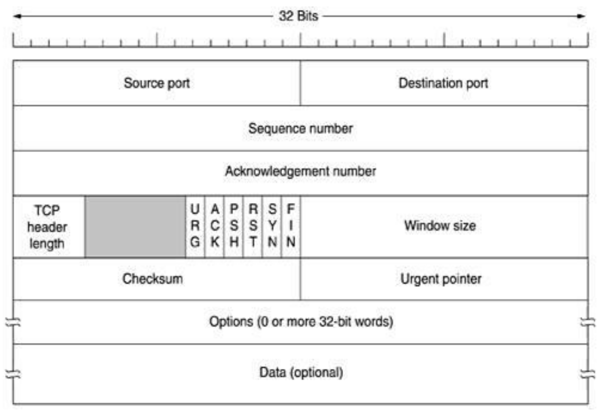


Fig 4.11: The TCP Header

- **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: I**t 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

- **R ST:** 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.

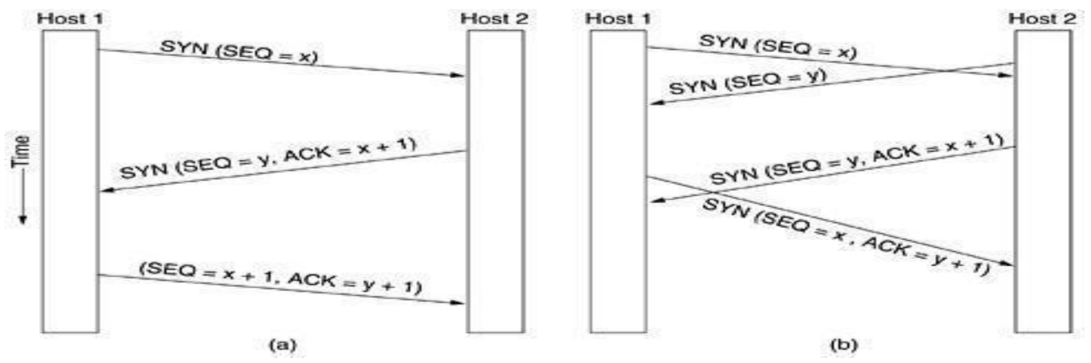

Host 1
Host 2
SYN (SEQ = x)
Time
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
(a)
Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x, ACK = y + 1)
(b)

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.

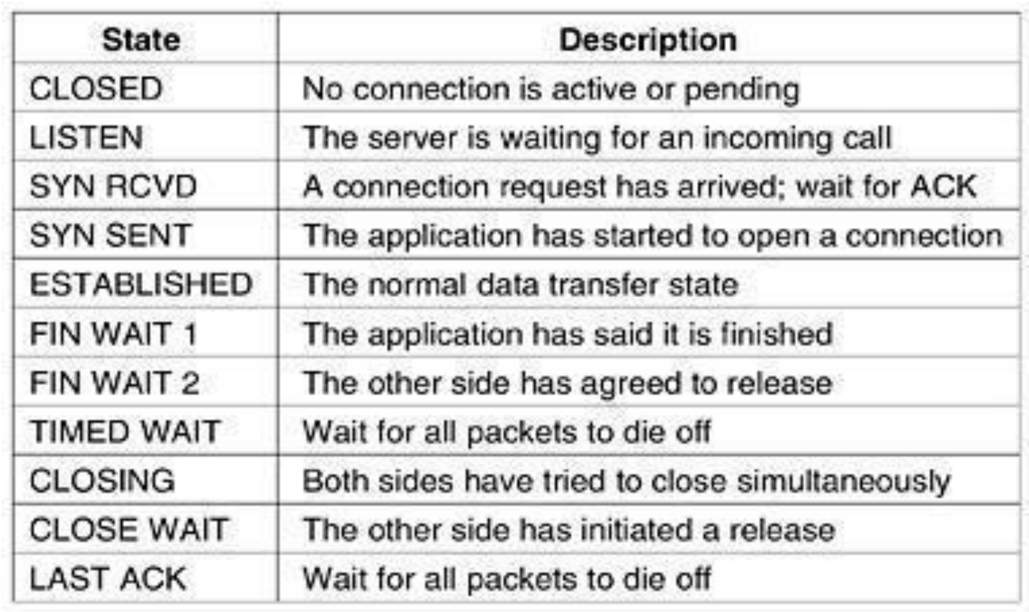
- When the *FIN* is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the other direction, however.

- 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.

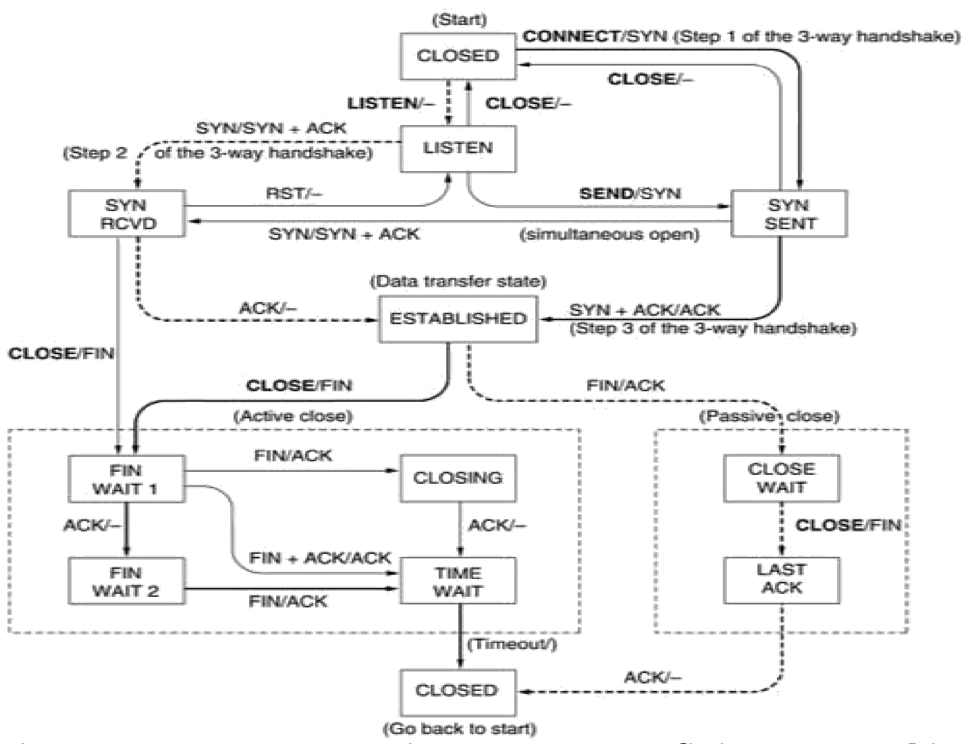


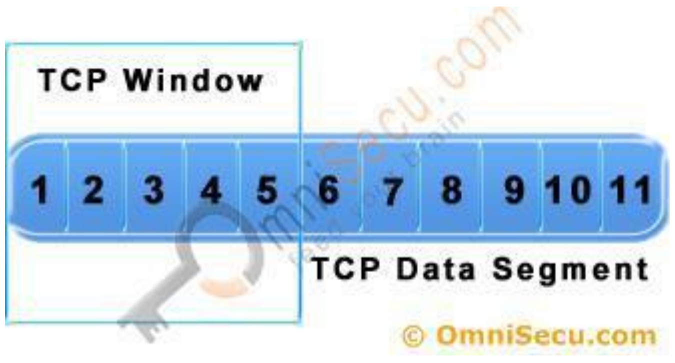
Figure 4.14 - TCP connection management finite state machine.

TCP Sliding Window

The working of the TCP sliding window mechanism can be explained as below.

The sending device can send all packets within the TCP window size (as specified in the TCP header) without receiving an ACK, and should start a timeout timer for each of them.

The receiving device should acknowledge each packet it received, indicating the sequence number of the last well-received packet. After receiving the ACK from the receiving device, the sending device slides the window to right side.



In this case, the sending device can send up to 5 TCP Segments without receiving an acknowledgement from the receiving device. After receiving the acknowledgement for Segment 1 from the receiving device, the sending device can slide its window one TCP Segment to the right side and the sending device can transmit segment 6 also.

If any TCP Segment lost while its journey to the destination, the receiving device cannot acknowledge the sender. Consider while transmission, all other Segments reached the destination except Segment 3. The receiving device can acknowledge up to Segment 2. At the sending device, a timeout will occur and it will re-transmit the lost Segment 3. Now the receiving device has received all the Segments, since only Segment 3 was lost. Now the receiving device will send the ACK for Segment 5, because it has received all the Segments to Segment 5.

Acknowledgement (ACK) for Segment 5 ensures the sender the receiver has succesfully received all the Segments up to 5.

TCP uses a byte level numbering system for communication. If the sequence number for a TCP segment at any instance was 5000 and the Segment carry 500 bytes, the sequence number for the next Segment will be 5000+500+1. That means TCP segment only

1. Congestion Avoidance Phase: After reaching

Slow Start Phase : exponential increment – In this phase after every RTT the congestion window size increments exponentially. Initially cwnd = 1

After 1 RTT, cwnd = 2^(1) = 2

2 RTT, cwnd = 2^(2) = 4

3 RTT, cwnd = 2^(3) = 8

Congestion Avoidance Phase : additive increment – This phase starts after the threshold value also denoted as *ssthresh*. The size of *cwnd*(congestion window) increases additive. After each RTT cwnd = cwnd + 1. Initially cwnd = i

After 1 RTT, cwnd = i+1

2 RTT, cwnd = i+2

3 RTT, cwnd = i+3

Congestion Detection Phase : multiplicative decrement – If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet which is assumed to two cases: when the RTO timer times out or when three duplicate ACKs are received.

- **Case 1 : Retransmission due to Timeout –** In this case congestion possibility is high.
  (a) ssthresh is reduced to half of the current window size.
  (b) set cwnd = 1
  (c) start with slow start phase again.

- **Case 2 : Retransmission due to 3 Acknowledgement Duplicates –** In this case congestion possibility is less.
  (a) ssthresh value reduces to half of the current window size.
  (b) set cwnd= ssthresh
  (c) start with congestion avoidance phase

The Features of TCP

1**.Connection oriented**: An application requests a ―connection‖ to destination and uses connection to transfer data

2**. Stream Data transfer**:- It is the duty of TCP to pack this byte stream to packets, known as TCP segments, which are passed to the IP layer for transmission to the destination device.
3. **Reliable:-** It recovers data from Network layer if data is damaged, duplicated or corrupted.
4. **Point to Point:-** TCP connection provides end t

Application Layer

Introduction

The **Application layer** provides services that directly support user applications, such as database access, e-mail, and file transfers.

It also allows applications to communicate with applications on other computers as though they were on the same computer.

When an application program uses network services, this is the layer it will access. For example, a web browser uses the Application layer to make requests for files and web pages; the Application layer then passes those requests down the stack, with each succeeding layer carrying out its specified task.

Application Layer Services

1. **Mail Services:** This layer provides the basis for E-mail forwarding and storage.

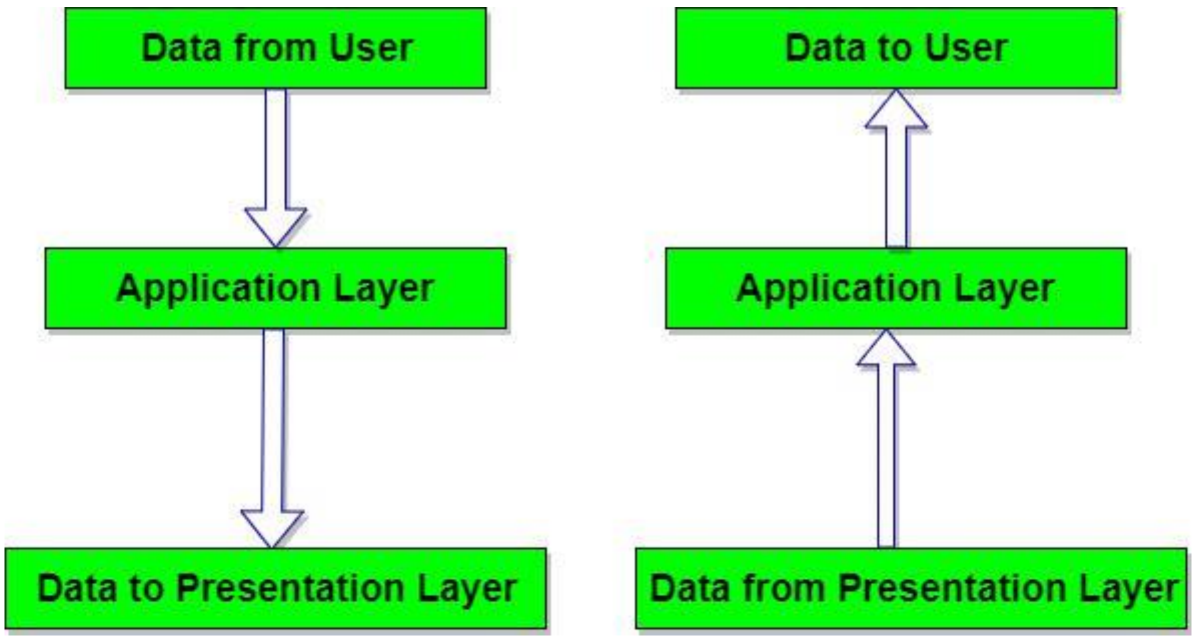
2. **Network Virtual Terminal:** It allows a user to log on to a remote host.

The application creates software emulation of a terminal at the remote host. User's computer talks to the software terminal which in turn talks to the host and vice versa. Then the remote host believes it is communicating with one of its own terminals and allows user to log on.

3. **Directory Services:** This layer provides access for global

information about various services.

4. **File Transfer, Access and Management (FTAM):** It is a standard mechanism to access files and manages it. Users can access files in a remote computer and manage it. They can also retrieve files from a remote computer.



Design Issues with Application Layer

There are commonly reoccurring problems that occur in the design and implementation of Application Layer protocols and can be addressed by patterns from several different pattern languages:

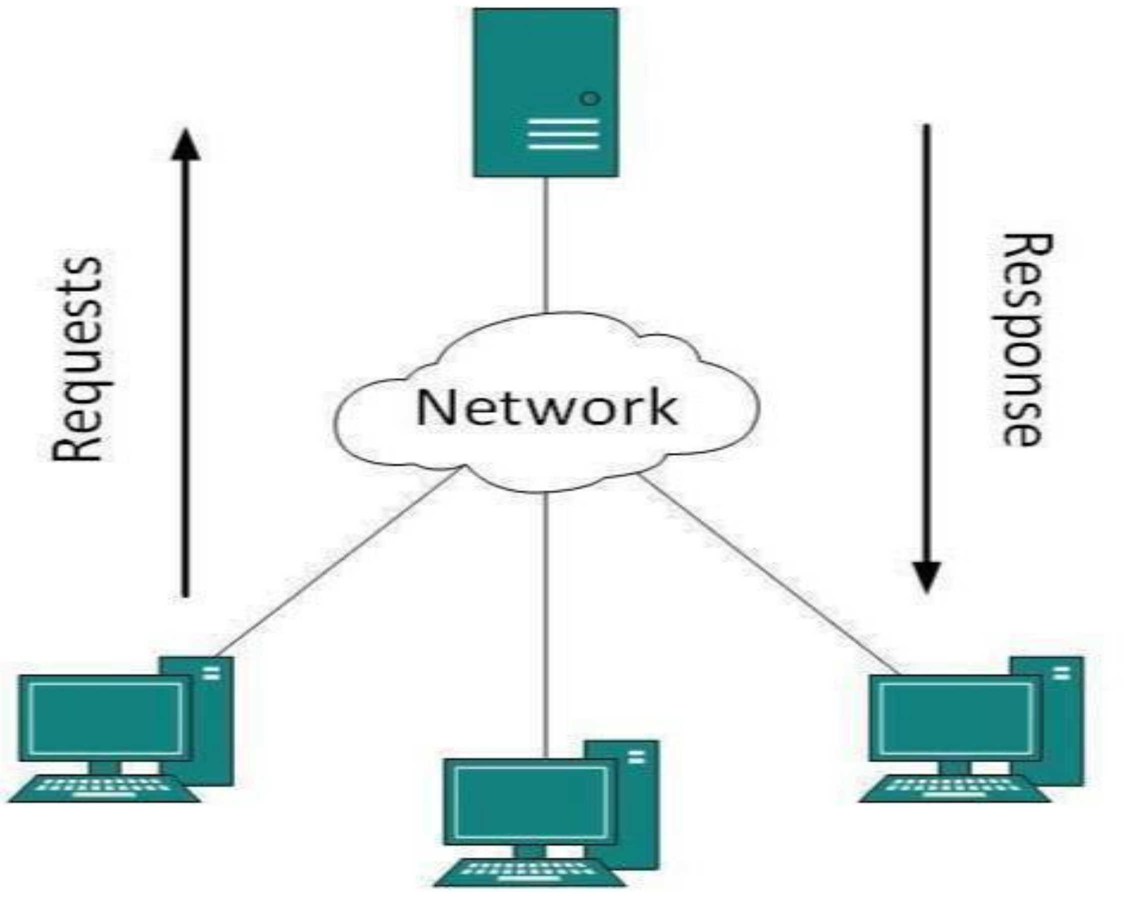
Application Layer Paradisms:

Client-Server Model:

Two remote application processes can communicate mainly in two different fashions:

- **Peer-to-peer:** Both remote processes are executing at same level and they exchange data using some shared resource.
- **Client-Server:** One remote process acts as a Client and requests some resource from another application process acting as Server.

In client-server model, any process can act as Server or Client. It is not the type of machine, size of the machine, or its computing power which makes it server; it is the ability of serving request that makes a machine a server.



A system can act as Server and Client simultaneously. That is, one process is acting as Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

Hyper Text Transfer Protocol (HTTP)

The Hyper Text Transfer Protocol (HTTP) is the foundation of World Wide Web. Hypertext is well organized documentation system which uses hyperlinks to link the pages in the text documents. HTTP works on client server model. When a user wants to access any HTTP page on the internet, the client machine at user end initiates a TCP connection to server on port 80. When the server accepts the client request, the client is authorized to access web pages.

To access the web pages, a client normally uses web browsers, who are responsible for initiating, maintaining, and closing TCP connections. HTTP is a stateless protocol, which means the Server maintains no information about earlier requests by clients.

HTTP versions

- HTTP 1.0 uses non persistent HTTP. At most one object can be sent over a single TCP connection.
- HTTP 1.1 uses persistent HTTP. In this version, multiple objects can be sent over a single TCP connection.

File Transfer Protocol(FTP)

The File Transfer Protocol (FTP) is the most widely used protocol for file transfer over the network. FTP uses TCP/IP for communication and it works on TCP port 21. FTP works on Client/Server Model where a client requests file from Server and server sends requested resource back to the client.

FTP uses out-of-band controlling i.e. FTP uses TCP port 20 for exchanging controlling information and the actual data is sent over TCP port 21.

The client requests the server for a file. When the server receives a request for a file, it opens a TCP connection for the client and transfers the file. After the transfer is complete, the server closes the connection. For a second file, client requests again and the server reopens a new TCP connection.

Domain Name System(DNS)

The Domain Name System (DNS) works on Client Server model. It uses UDP protocol for transport layer communication. DNS uses hierarchical

domain based naming scheme. The DNS server is configured with Fully Qualified Domain Names (FQDN) and email addresses mapped with their respective Internet Protocol addresses.

A DNS server is requested with FQDN and it responds back with the IP address mapped with it. DNS uses UDP port 53.

Simple Mail Transfer Protocol(SMTP)

The Simple Mail Transfer Protocol (SMTP) is used to transfer electronic mail from one user to another. This task is done by means of email client software (User Agents) the user is using. User Agents help the user to type and format the email and store it until internet is available.

When an email is submitted to send, the sending process is handled by Message Transfer Agent which is normally comes inbuilt in email client software.

Message Transfer Agent uses SMTP to forward the email to another Message Transfer Agent (Server side). While SMTP is used by end user to only send the emails, the Servers normally use SMTP to send as well as receive emails. SMTP uses TCP port number 25 and 587.

Client software uses Internet Message Access Protocol (IMAP) or POP protocols to receive emails.

Telnet

Telnet is a network protocol that allows a user to communicate with a remote device. It is a virtual terminal protocol used mostly by network administrators to remotely access and manage devices. Administrator can access the device by *telnetting* to the IP address or hostname of a remote device.

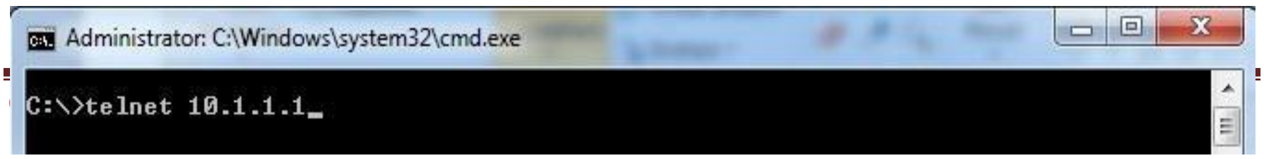
To use telnet, you must have a software (Telnet client) installed. On a remote device, a Telnet server must be installed and running. Telnet uses the TCP port 23 by default.

One of the greatest disadvantages of this protocol is that all data, including usernames and passwords, is sent in clear text, which is a potential security risk. This is the main reason why Telnet is rarely used today and is being replaced by a much secure protocol called SSH. Here you can find information about setting up Telnet access on your Cisco device.

NOTE

The word **telnet** can also refer to the software that implements the telnet protocol.

On Windows, you can start a Telnet session by typing the *telnet IP_ADDRESS or HOSTNAME* command:



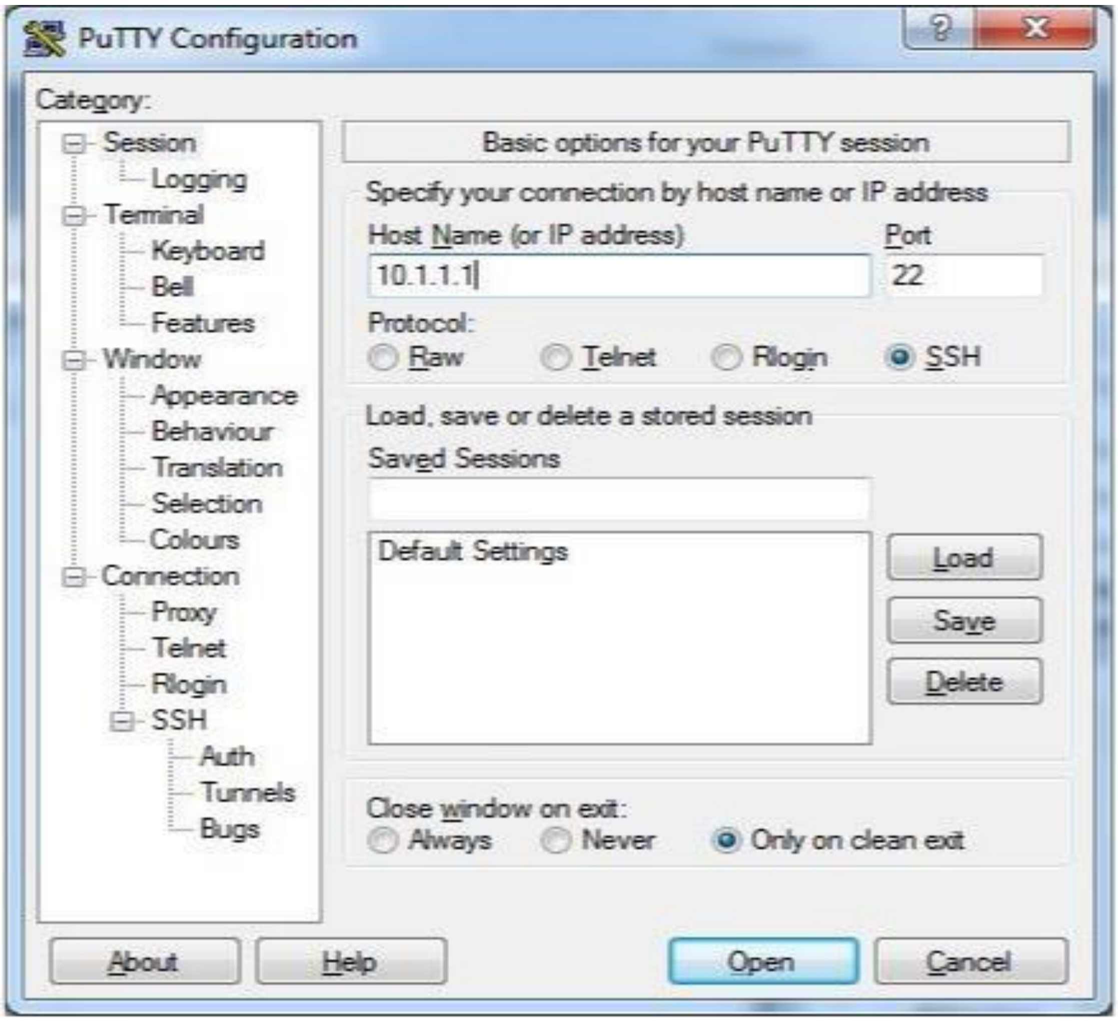
```
C:\>telnet 10.1.1.1_
```

SSH (Secure Shell)

SSH is a network protocol used to remotely access and manage a device. The key difference between Telnet and SSH is that SSH uses encryption, which means that all data transmitted over a network is secure from eavesdropping. SSH uses the **public key encryption** for such purposes.

Like Telnet, a user accessing a remote device must have an SSH client installed. On a remote device, an SSH server must be installed and running. SSH uses the TCP port 22 by default.

Here is an example of creating an SSH session using Putty, a free SSH client:



PuTTY Configuration

Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address): 10.1.1.1

Port: 22

Protocol: Raw Telnet Rlogin [✓ SSH]

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

NOTE

SSH is the most common way to remotely access and manage a Cisco setting up SSH access on your Cisco device.