DIGITAL NOTES ON

COMPUTER NETWORKS

R20A0510

B.TECH IIIYEAR – I SEM (R20) (2023-24)

PREPARED BY

A DILEEP

DEPARTMENT OF COMPUTER SCIENCE(D.S)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified) Maisammaguda, Dhulapally(Post Via. Hakimpet), Secunderabad – 500100, Telangana State, INDIA.

MALLA REDDY COLLEGE OF ENGINEERING AND COLLEGE
B.TECH - III– YEAR I-SEM-D.S

L/T/P/C
3/-/-/3

(R20A0510) COMPUTER NETWORKS

COURSE OBJECTIVES:

1. To understand the fundamentals of computer networks, TCP/IP & OSI model.
2. To know Data link layer Issues & Protocols.
3. To learn Network layer Protocols & IP addressing concepts.
4. To identify end to end communication & various aspects of Transport layer.
5. To understand various user services in a network.

UNIT - I:

Introduction: Network, Uses of Networks, Types of Networks, And Reference Models: TCP/IP Model, The OSI Model, Comparison of the OSI and TCP/IP reference model.

Physical Layer: Guided transmission media, Wireless transmission media, Switching.

UNIT - II:

Data Link Layer - Design issues, Error Detection & Correction, Elementary Data Link Layer Protocols, Sliding window protocols.

Multiple Access Protocols - ALOHA, CSMA, CSMA/CD, CSMA/CA, Collision free protocols, Ethernet-Physical Layer & Mac Sub layer.

UNIT - III:

Network Layer: Network Layer Design issues, Store and Forward Packet Switching Connection less and Connection-oriented networks, Routing algorithms: Optimality principle, Shortest path, Flooding, Distance Vector Routing, Count to Infinity Problem, Link State Routing, Path Vector Routing, Hierarchical Routing; Congestion control algorithms, IP addresses, CIDR, Sub netting, Super Netting, IPv4, Packet Fragmentation, IPv6 Protocol, Transition from IPv4 to IPv6, ARP, RARP.

UNIT - IV:

Transport Layer: Services provided to the upper layers, Elements of transport protocol, Addressing,Connection Establishment, Connection Release, Error Control & Flow Control, Crash Recovery.

The Internet Transport Protocols: UDP, Introduction to TCP, The TCP Service Model,The TCP Segment Header, The Connection Establishment, The TCP Connection Release, The TCP Sliding Window, The TCP Congestion Control Algorithm.

UNIT - V:

Application Layer- Introduction, Providing services, Applications layer paradigms: Client server model, HTTP, E-mail, WWW, TELNET, DNS.

TEXT BOOKS:

1. Computer Networks - Andrew S Tanenbaum, 4th Edition, Pearson Education.
2. Data Communications and Networking - Behrouz A. Forouzan, Fifth Edition TMH,2013.

REFERENCES BOOKS:

1. An Engineering Approach to Computer Networks - S. Keshav, 2nd Edition,Pearson Education.
2. Understanding communications and Networks, 3rd Edition, W. A. Shay, Cengage Learning.
3. Computer Networking: A Top-Down Approach Featuring the Internet, James F.Kurose, K.W. Ross, 3rd Edition, Pearson Education.

COURSE OUTCOMES:

At the end of this course, students will be able to:

- Describe basics of Computer Networks and Reference Models.
- Apply the Datalink Layer Concepts.
- Know allotment of IP addresses, best routing path calculations in network.
- Analyze TCP, UDP working and know how to handle congestion
- Observe & Apply in Application Layer issues in various internet services.

INDEX

UNIT-1

INTRODUCTION: -

An interconnected collection of autonomous computers is called a **computer network**. Two computers are said to be interconnected if they are able to exchange the information. If one computer can forcibly start, stop and control another one, the computers are not autonomous. A system with one control unit and many slaves is not a network, nor is a large computer with remote printers and terminals. A network is a set of devices connected by media links. Anode can be a computer, printer or any other device capable of sending and receiving data generated by other nodes on the network. The links connecting the devices are often called communication channels.

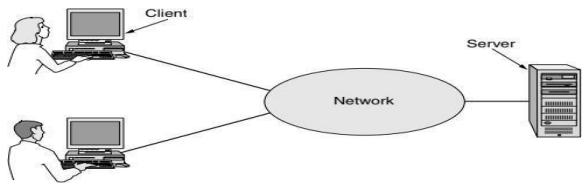
1.1. USES OF NETWORKS: -

Business Applications

Most companies have a substantial number of computers. For example, a company may have a computer for each worker and use them to design products, write brochures, and do the payroll. Initially, some of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to distribute information throughout the company. Put in slightly more general form, the issue here is resource sharing. The goal is to make all programs, equipment, and especially data available to anyone on the network without regard to the physical location of the resource or the user.

Networks called VPNs (Virtual Private Networks) may be used to join the individual networks at different sites into one extended network. In other words, the mere fact that a user happens to be 15,000 km away from his data should not prevent him from using the data as though they were local. The data are stored on powerful computers called servers. Often these are centrally housed and maintained by a system administrator. In contrast, the employees have simpler machines, called clients, on their desks, with which they access remote data. This whole arrangement is called the client-server model. It is widely used and forms the basis of much network usage. The most popular realization is that of a Web application, in which the server generates Web pages based on its database in response to client requests that may update the database.

Goal for many companies is doing business electronically, especially with customers and suppliers. This new model is called e-commerce (electronic commerce) and it has grown rapidly in recent years. Airlines, bookstores, and other retailers have discovered that many customers like the convenience of shopping from home.



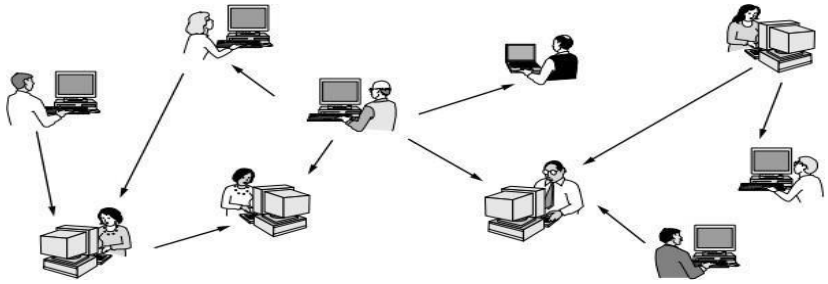
1-1 A NETWORK WITH TWO CLIENTS AND ONE SERVER

Home Applications: -

Now, many consumer electronic devices, such as set-top boxes, game consoles, and clock radios, come with embedded computers and computer networks, especially wireless networks, and home networks are broadly used for entertainment, including listening to, looking at, and creating music, photos, and videos. Internet access provides home users with

connectivity to remote computers. As with companies, home users can access information, communicate with other people, and buy products and services with e-commerce.

Much of the information is accessed using the client-server model, but there is different, popular model for accessing information that goes by the name of peer-to-peer communication. In this form, individuals who form a loose group can communicate with others in the group, as shown in Figure



1-2 IN A PEER TO PEER SYSTEM THAT ARE NO FIXED CLIENTS AND SERVERS

Peer-to-peer communication is often used to share music and videos. These include fans sharing public domain music, families sharing photos and movies, and users downloading public software packages. In fact, one of the most popular Internet applications of all, email, is inherently peer-to-peer. There are multi-person messaging services too, such as the Twitter service that lets people send short text messages called ‘‘tweets’’ to their circle of friends or other willing audiences.

Between person-to-person communications and accessing information are social network applications like Facebook which lets people update their personal profiles and shares the updates with other people who they have declared to be their friends.

Ubiquitous computing, in which computing is embedded into everyday life, many homes are already wired with security systems that include door and window sensors, and there are many more sensors that can be folded in to a smart home monitor, such as energy consumption. Your electricity, gas and water meters could also report usage over the network. This would save money as there would be no need to send out meter readers.

Devices such as televisions that plug into the wall can use power-line networks to send information throughout the house over the wires that carry electricity.

A technology called RFID (Radio Frequency Identification) will push this idea even further in the future. RFID tags are passive (i.e., have no battery) chips the size of stamps and they can already be affixed to books, passports, pets, credit cards, and other items in the home and out. This lets RFID readers locate and communicate with the items over a distance of up to several meters, depending on the kind of RFID.

Mobile Users :-

Mobile computers, such as laptop and handheld computers, are one of the fastest-growing segments of the computer industry. Connectivity to the Internet enables many of these mobile uses. Since having a wired connection is impossible in cars, boats, and airplanes, there is a lot of interest in wireless networks. Cellular networks operated by the telephone companies are one familiar kind of wireless network that blankets us with coverage for mobile phones. Wireless hotspots based on the 802.11 standard are another kind of wireless network for mobile computers.

Perhaps the key driver of mobile, wireless applications is the mobile phone. Text messaging or texting is tremendously popular. It lets a mobile phone user type a short message that is then delivered by the cellular network to another mobile subscriber. Smart phones, such as the popular iPhone, combine aspects of mobile phones and mobile computers.

Since mobile phones know their locations, often because they are equipped with GPS (Global Positioning System) receivers, some services are intentionally location dependent. Mobile maps and directions are an obvious candidate as your GPS-enabled phone and car probably have a better idea of where you are than you do.

SOCIAL ISSUES: -

Social networks, message boards, content sharing sites, and a host of other applications allow people to share their views with like-minded individuals. As long as the subjects are restricted to technical topics or hobbies like gardening, not too many problems will arise. The trouble comes with topics that people actually care about, like politics, religion, or sex. Views that are publicly posted may be deeply offensive to some people. Worse yet, they may not be politically correct. Furthermore, opinions need not be limited to text; high-resolution color photographs and video clips are easily shared over computer networks. Some people take a live-and-let-live view, but others feel that posting certain material (e.g., verbal attacks on particular countries or religions, pornography, etc.) is simply unacceptable and that such content must be censored. Different countries have different and conflicting laws in this area. Thus, the debate rages.

1.2. TYPES OF NETWORKS: -

There are two types of transmission technology that are in widespread use. They are as follows:

1. Broadcast links.

2. Point-to-point links.

Broadcast networks have a single communication channel that is shared by all the machines on the network. Short messages, called packets in certain contexts, sent by any machine are received by all the others.

point-to-point networks consist of many connections between individual pairs of machines.

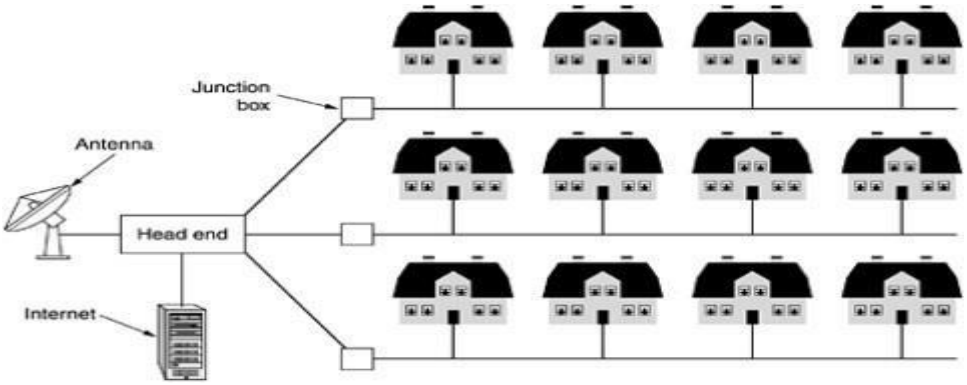
Local Area Networks: -

Local area networks, generally called LANs, are privately-owned networks within a single building or campus of up to a few kilometers in size. They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information. LANs are distinguished from other kinds of networks by three characteristics: (1) their size, (2) their transmission technology, and (3) their topology.

LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. Knowing this bound makes it possible to use certain kinds of designs that would not otherwise be possible. It also simplifies network management.

Metropolitan Area Networks: -

A metropolitan area network, or MAN, covers a city. The best-known example of a MAN is the cable television network available in many cities. This system grew from earlier community antenna systems used in areas with poor over-the-air television reception. In these early systems, a large antenna was placed on top of a nearby hill and signal was then piped to the subscribers' houses.



1-4: ***A metropolitan area network based on cable TV.***

Wide Area Networks :-

A wide area network, or WAN, spans a large geographical area, often a country or continent. It contains a collection of machines intended for running user (i.e., application) programs. We will follow traditional usage and call these machines hosts. The hosts are connected by a communication subnet, or just subnet for short. The hosts are owned by the customers (e.g., people's personal computers), whereas the communication subnet is typically owned and operated by a telephone company or Internet service provider. The job of the subnet is to carry messages from host to host, just as the telephone system carries words from speaker to listener.

Separation of the pure communication aspects of the network (the subnet) from the application aspects (the hosts), greatly simplifies the complete network design. In most wide area networks, the subnet consists of two distinct components: transmission lines and switching elements. Transmission lines move bits between machines. They can be made of copper wire, optical fiber, or even radio links. Switching elements are specialized computers that connect three or more transmission lines.

When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them.

1.3. REFERENCE MODELS: -

TCP/IP MODEL

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:

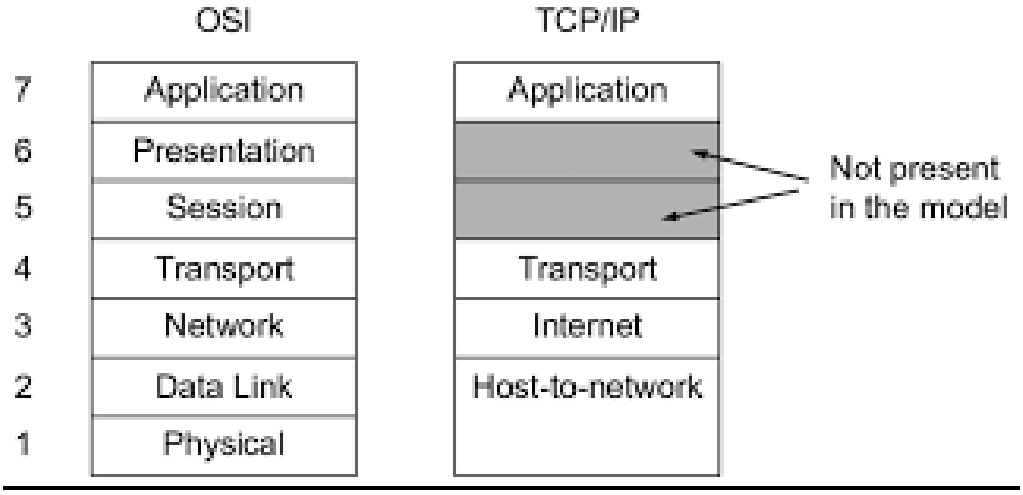
The TCP/IP reference model does not really say much about what happens here, except to point out that the host has to connect to the network using some protocol so it can send IP packets to it. This protocol is not defined and varies from host to host and network to network.

Internet Layer:

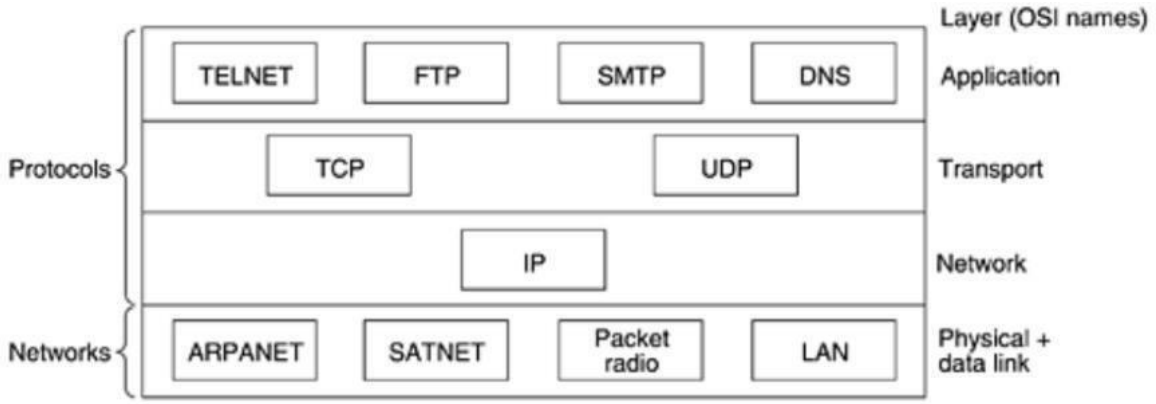
This layer, called the internet layer, is the linchpin that holds the whole architecture together. Its job is to permit hosts to inject packets into any network and have they travel independently to the destination (potentially on a different network). They may even arrive in a different order than they 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.



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.



1-6 PROTOCOLS AND NETWORKS IN THE TCP/IP MODEL INITIALLY

The Application Layer

The TCP/IP model does not have session or presentation layers. No need for them was perceived, so they were not included. Experience with the OSI model has proven this view correct: they are of little use to most applications.

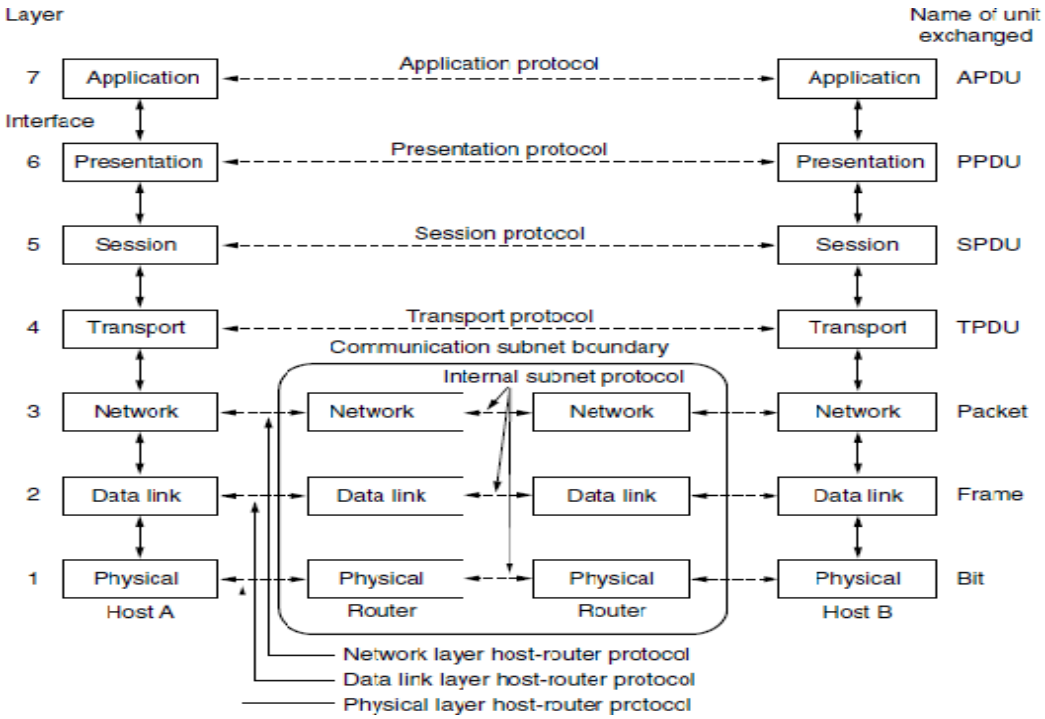
On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP), as shown in Fig.. The virtual terminal protocol allows a user on one machine to log onto a distant machine and work there. The file transfer protocol provides away 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.

The Host-to-Network Layer

Below the internet layer is a great void. 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. Books and papers about the TCP/IP model rarely discuss it.

The OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 1-20. 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. We will just call it the OSI model for short.



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.

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. Typical questions here are how many volts should be used to represent a 1 and how many for a 0, how many nanoseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established and how it is torn down when both sides are finished, and how many pins the network connector has and what each pin is used for. The design issues here largely deal with mechanical, electrical, and timing interfaces, and the physical transmission medium, which lies below the physical layer.

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 transmit 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.

Broadcast networks have an additional issue in the data link layer: how to control access to the shared channel. A special sublayer of the data link layer, the medium access control sublayer, deals with this problem.

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 messagesor 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. (As an aside, an error-free channel is impossible to achieve; what people really mean by this term is that the error rate is low enough to ignore in practice.)

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 neighbors, and not between the ultimate source and destination machines, which may be separated by many routers. The difference between layers 1 through 3, which are chained, and layers 4 through 7, which are end-to-end, is illustrated in Fig

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 (checkpointing long transmissions to allow them to continue from where they were after a crash).

The Presentation Layer

Unlike lower layers, which are mostly concerned with moving bits around, 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 (Hyper Text 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.

A Comparison of the OSI and TCP/IP Reference Models

The OSI and TCP/IP reference models have much in common. Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. For example, in both models the layers up through and including the transport layer are there to provide an end-to-end, network-independent transport service to processes wishing to communicate. These layers form the transport provider. Again, in both models, the layers above transport are application-oriented users of the transport service.

Despite these fundamental similarities, the two models also have many differences. In this section we will focus on the key differences between the two reference models. It is important to note that we are comparing the reference models here, not the corresponding protocol stacks. The protocols themselves will be discussed later.

For an entire book comparing and contrasting TCP/IP and OSI

Three concepts are central to the OSI model:

1. Services.

2. Interfaces.

3. Protocols.

Probably the biggest contribution of the OSI model is to make the distinction between these three concepts explicit. Each layer performs some services for the layer above it. The service definition tells what the layer does, not how entities above it access it or how the layer works. It defines the layer's semantics.

A layer's interface tells the processes above it how to access it. It specifies what the parameters are and what results to expect. It, too, says nothing about how the layer works inside.

Finally, the peer protocols used in a layer are the layer's own business. It can use any protocols it wants to, as long as it gets the job done (i.e., provides the offered services). It can also change them at will without affecting software in higher layers.

These ideas fit very nicely with modern ideas about object-oriented programming. An object, like a layer, has a set of methods (operations) that processes outside the object can invoke. The semantics of these methods define the set of services that the object offers. The methods' parameters and results form the object's interface.

The code internal to the object is its protocol and is not visible or of any concern outside the object.

The TCP/IP model did not originally clearly distinguish between service, interface, and protocol, although people have tried to retrofit it after the fact to make it more OSI-like. For example, the only real services offered by the internet layer are SEND IP PACKET and RECEIVE IP PACKET.

As a consequence, the protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes. Being able to make such changes is one of the main purposes of having layered protocols in the first place.

The OSI reference model was devised before the corresponding protocols were invented. This ordering means that the model was not biased toward one particular set of protocols, a fact that made it quite general. The downside of this ordering is that the designers did not have much experience with the subject and did not have a good idea of which functionality to put in which layer.
For example, the data link layer originally dealt only with point-to-point networks. When broadcast networks came around, a new sublayer had to be hacked into the model. When people started to build real networks using the OSI model and existing protocols, it was discovered that these networks did not match the required service specifications (wonder of wonders), so convergence sublayers had to be grafted onto the model to provide a place for papering over thedifferences. Finally, the committee originally expected that each country would have one network, run by the government and using the OSI protocols, so no thought was given to internetworking.

With TCP/IP the reverse was true: the protocols came first, and the model was really just a description of the existing protocols. There was no problem with the protocols fitting the model. They fit perfectly. The only trouble was that the model did not fit any other protocol stacks. Consequently, it was not especially useful for describing other, non-TCP/IP networks.

Turning from philosophical matters to more specific ones, an obvious difference between the two models is the number of layers: the OSI model has seven layers and the TCP/IP has four layers. Both have (inter)network, transport, and application layers, but the other layers are different.

Another difference is in the area of connectionless versus connection-oriented communication. The OSI model supports both connectionless and connection-oriented communication in the network layer, but only connection-oriented communication in the transport layer, where it counts (because the transport service is visible to the users). The TCP/IP model has only one mode in the network layer (connectionless) but supports both modes in the transport layer, giving the users a choice. This choice is especially important for simple request-response protocols.

PHYSICAL LAYER: -

GUIDED TRANSMISSION 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.

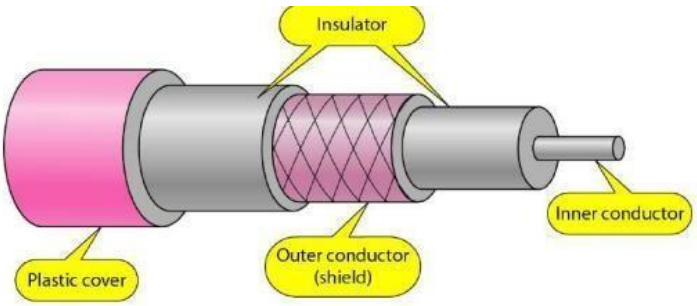
1. Twisted-Pair Cable A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together, as shown below figure.

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 twistedpair cables. Local-area networks, such as lOBase-T and lOOBase-T, also use twisted- pair cables.

2. Coaxial Cable Coaxial cable (or coax)

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).

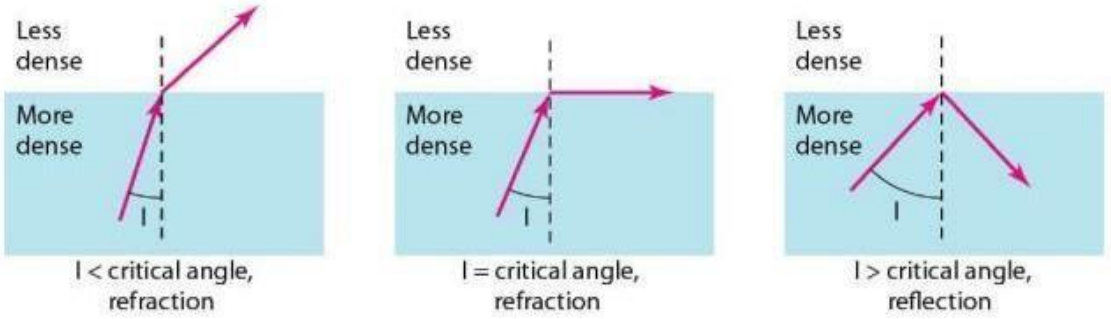


Applications

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,

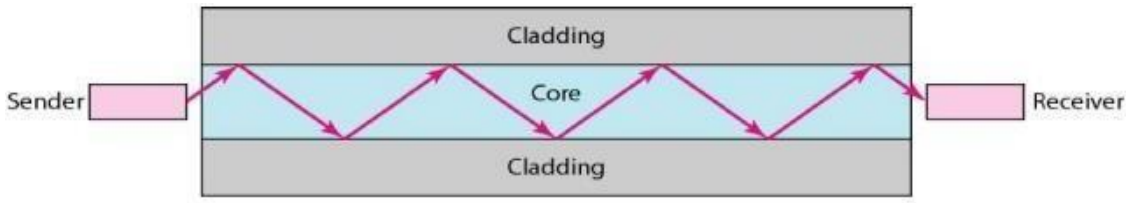
however, cable TV providers replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable. Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high 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 shows how a ray of light changes direction when going from a more dense to a less dense substance.

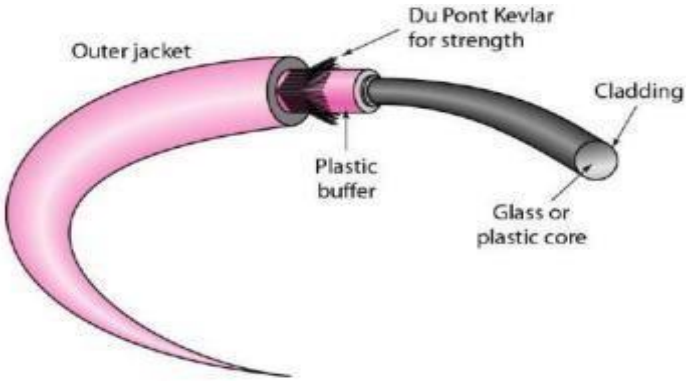


As the figure shows, if the angle of incidence I (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the critical angle, the ray refracts and moves closer to the surface. If the angle ofincidence is equal to the critical angle, the light bends 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.



Cable Composition Figure shows the composition of a typical fiber-optic cable. The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable. Kevlar is a strong material used in the fabrication of bulletproof vests. Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.



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 fiberoptic 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.

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

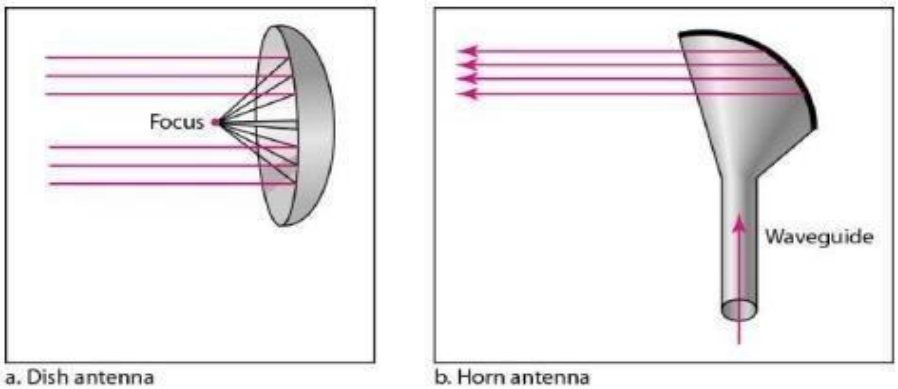
3. The microwave band is relatively wide, almost 299 GHz. Therefore wider sub bands can be assigned, and a high data rate is possible

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.



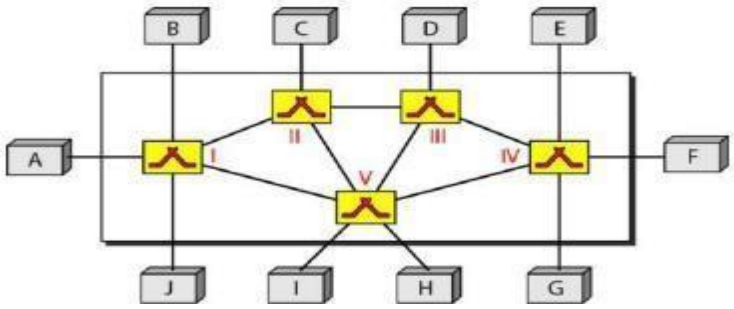
a. Dish antenna b. Horn antenna

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.

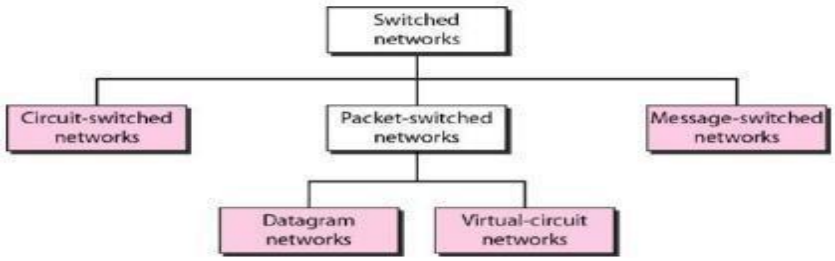
Applications The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate. The Infrared Data Association (IrDA), an association for sponsoring the use of infrared waves, has established standards 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.

SWITCHING: -

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology).These methods, however, are impractical and wasteful when applied to very large networks. The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most ofthe time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment. A better solution is switching. A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure 8.1 shows a switched network

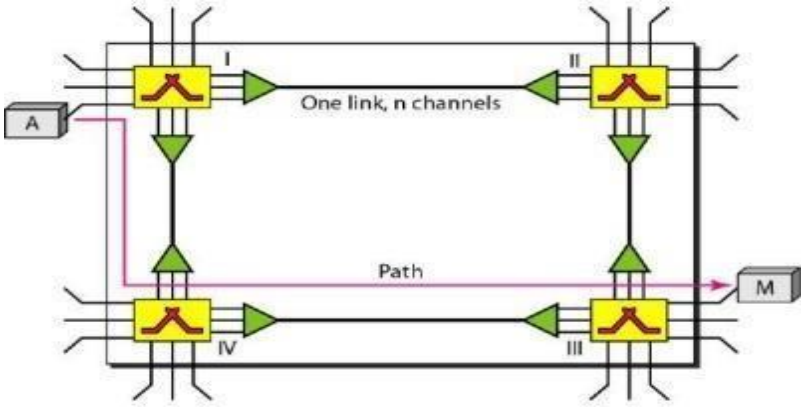


The end systems (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links. Traditionally, three methods of switching have been important: circuit switching, packet switching, and message switching. The first two are commonly used today. The third has been phased out in general communications but still has networking applications. We can then divide today's networks into three broad categories: circuit-switched networks, packet-switched networks, and message switched.



CIRCUIT-SWITCHED NETWORKS

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link. Each link is normally divided into n channels by using FDM or TDM. Figure 8.3 shows a trivial circuit-switched network with four switches and four links. Each link is divided into n (n is 3 in the figure) channels by using FDM or TDM. The end systems, such as computers or telephones, are directly connected to a switch. We have shown only two end systems for simplicity. When end system A needs to communicate with end system M, system A needs to request a connection to M that must be accepted by all switches as well as by M itself. This is called the setup phase; a circuit (channel) is reserved on each link, and the combination of circuits or channels defines the dedicated path. After the dedicated path made of connected circuits (channels) is established, data transfer can take place. After all data have been transferred, the circuits are torn down.



Three Phases The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Setup Phase Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches. For example, in Figure , when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time. In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established.

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

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

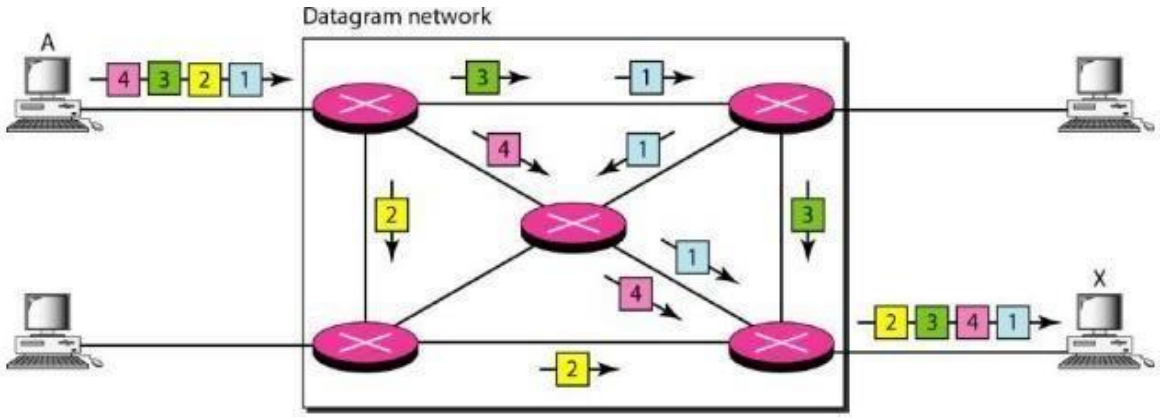
1. PACKET SWITCHED NETWORK In a Computer Network, the communication between two ends is done in blocks of data called packets. So instead of continuous communication the exchange takes place in the form of individual packets between the two computers. This allows us to make the switches function for both storing and forwarding because a packet is an independent entity that can be stored and sent later.

a. DATAGRAM NETWORKS

In data communications, we need to send messages from one end system to another. If the message is going to pass through a packet-switched network, it needs to be divided into packets of fixed or variable size. The size of the packet is determined by the network and the governing protocol. In packet switching, there is no resource allocation for a packet.

This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. The allocation is done on a first come, first-served basis. When a switch receives a packet, no matter what is the source or destination, the packet must wait if there are other packets being processed. As with other systems in our daily life, this lack of reservation may create delay.

For example, if we do not have a reservation at a restaurant, we might have to wait. In a datagram network, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as datagrams. Datagram switching is normally done at the network layer. The switches in a datagram network are traditionally referred to as routers.



In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X. This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between them packets. Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upperlayer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application. The datagram networks are sometimes referred to as connectionless networks. The term connectionless here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

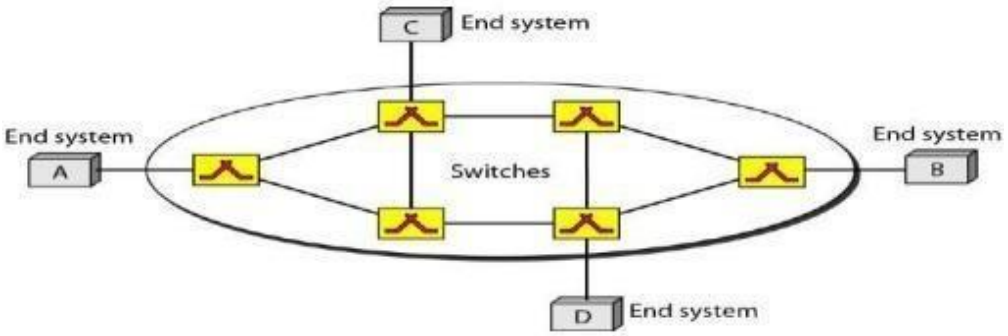
b. VIRTUAL-CIRCUIT NETWORKS

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

- As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
- Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
- As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction, not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet.

  As in a circuit-switched network, all packets follow the same path established during the connection.
- A virtual-circuit network is normally implemented in the data link layer, while a circuit switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.



UNIT –II

Introduction

The Data Link Layer breaks the bit stream into discrete frames and computes the checksum for each frame. When a Frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from one computed contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

DATA LINK LAYER DESIGN ISSUES

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

1. Providing a well-defined service interface to the network layer.

2. Dealing with transmission errors.

3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig. 2-1. Frame management forms the heart of what the data link layer does. In the following sections we will examine all the above mentioned issues in detail.

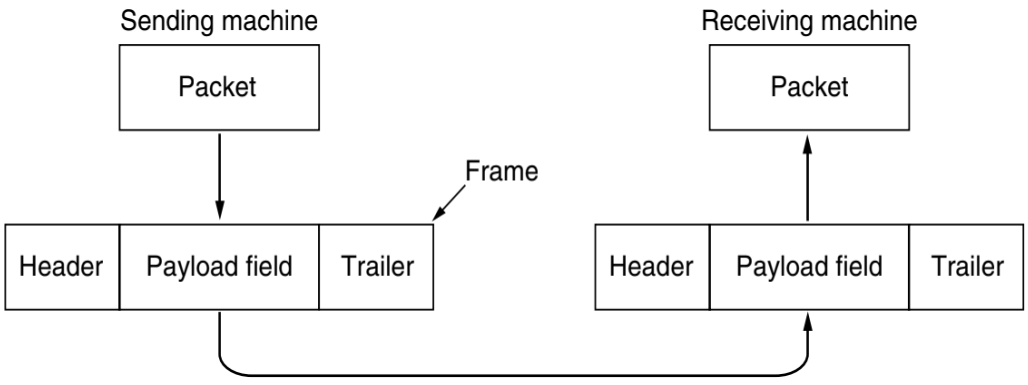


Figure 2-1. Relationship between packets and frames.

Services Provided 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. On the source machine is an entity, call it a process, in the network layer that hands some bits to the data link layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the net-work layer there, as shown in Fig. 2-2(a). The actual transmission follows the path of Fig. 2-2(b), but it is easier to think in terms of two data link layer processes communicating using a data link protocol. For this reason, we will implicitly use the model of Fig. 2-2(a) throughout this chapter.

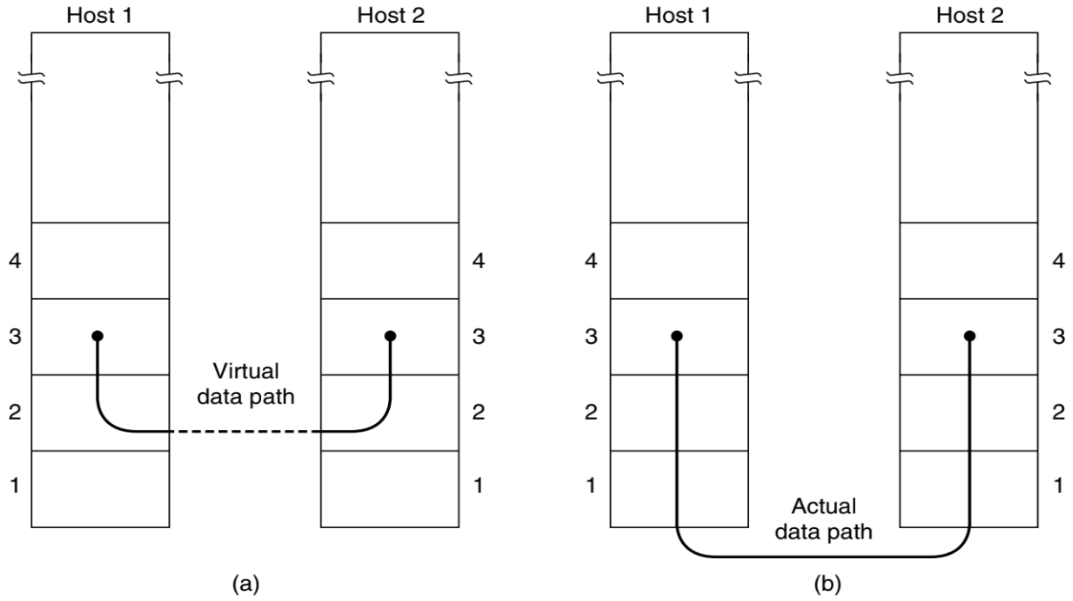


Figure 2-2. (a) Virtual communication. (b) Actual communication.

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

1. Unacknowledged connectionless service.
   Ethernet is a good example of a data link layer that provides this class of service. This service is appropriate when the error rate is very low, It is also appropriate for real-time traffic, such as voice, in which late data are worse than bad data.

2. Acknowledged connectionless service.
   This service is useful over unreliable channels, such as wireless systems. 802.11 (WiFi) is a good example of this class of service.

3. Acknowledged connection-oriented

It is appropriate over long, unreliable links such as a satellite channel or a long-distance telephone circuit.

The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted. When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report).

Breaking up the bit stream into frames is more difficult than it at first appears. A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at four methods:

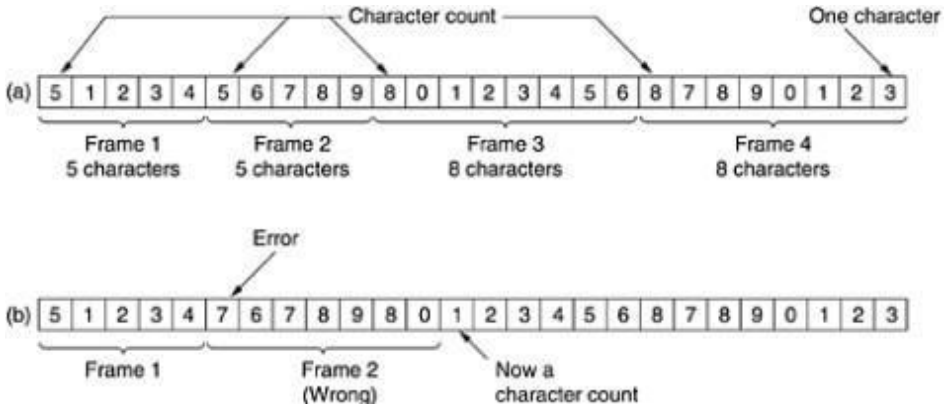
FRAMING METHODS

1. CHARATER COUNT METHOD
2. STARTING AND ENDING CHARACTERS, WITH CHARATER STUFFING
3. STARTING AND ENDING FLAGS, WITH BIT STUFFING

CHARATER COUNT METHOD:

In this method a field in the header will be used to specify the number of CHARACTERS in the frame. When data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.

The trouble with this algorithm is that the count can be garbed by a transmission error resulting the destination will get out of synchronization and will be unable to locate the start of the next frame. There is no way of telling where the next frame starts. For this reason this method is rarely used.



A Character Stream (a) Without errors (b) With one error

CHARATER STUFFING METHOD:In this method each frame will start with a FLAG and ends with a FLAG.

The starting flag is **DLE STX ---- Data Link Escape Start of Text**

The ending flag is **DLE ETX ------ Data link Escape End of Text.**

Ex 1. The given Data **ABRFCXDGJHKK12435ASBGXRR**
The Data will be sent as **DLE STX ABRFCXDGJHKK12435ASBGXRR DLE STX**

Ex 2. The given Data **ASHGTRDXZBNHG DLE STX %$#54378**

The data will be sent as **DLE STX ASHGTRDXZBNHG DLE STX %$#54378 DLE ETX**

Dis Adv:

1. **24 bits are unnecessarily stuffed.**
2. **Transmission delay.**

BIT STUFFING METHOD

Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte).Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag bytein the data. When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e.,deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110. Figure 3-6 gives an example of bit stuffing.

ERROR – CORRECTING AND DETECTING CODES

Network designers have developed two basics strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been .The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred,but not which error and have it request a retransmission. The former strategy uses **error**

– correcting codes and the latter uses **error- detecting codes**. The use of error-correcting
– codes is often referred to as FEC (Forward Error Correction).

The **error – correcting and error- detecting methods are**

1. PARITY METHOD
2. LRC METHOD (Longitudinal redundancy check)
3. CRC METHOD (Cyclic redundancy check)
4. HAMMING CODE METHOD

PARITY METHOD

- appends a parity bit to the end of each word in the frame
- Even parity is used for asynchronous Transmission
- Odd parity is used for synchronous Transmission

S.No.	Character code	even parity	odd parity
Ex 1	1100100	1100100 1	1100100 0
Ex 2	0011000	0011000 0	0011000 1

If one bit or any odd no. of bits is erroneously inverted during Transmission, the Receiver will detect an error. However if two or even no. of bits are inverted an undetected error occurs.

Ex 3. The Transmitted data is 10011010. The received data is 11011010. Let both the transmitter and receiver are agreed on EVEN parity. Now an error will be detected, since the no of one's received are ODD

4. The Transmitted data is 10011010. The received data is 01011010 the received data is wrong even though the no of ones are EVEN. Science two bits are inverted error can't be detected

Longitudinal Redundancy Check (LRC)

The frame is viewed as a block of characters arranged in 2-dimensions. To each character is appended a parity bit. In addition a parity bit is generated for each bit position across all characters i.e., an additional character is generated in which the I^{th} bit of the character is parity bit for the I^{th} bit of all other characters in the block. This can be expressed mathematically using exclusive OR(+) operation. The parity bit at the end of each character of row parity

$$R_j = b_{1j} + b_{2j} + \text{--------} b_{nj}$$

Where R_j=Parity bit of jth character

b_{ij}=ith bit in jth character

This equation generates even parity.

$$C_i = b_{i1} + b_{i2} + \text{----} + b_{in}$$

Where C_i=ith bit of parity check character

m=number of characters in a frame

In this format the parity bits at the end of each character are referred to as

The Vertical Redundancy Check (VRC) and the Parity check character is referred to as the Longitudinal Redundancy Check (LRC).

	bit 1	bit 2	bit n	Parity bit	
					VRC
Character 1	b11	b21	bn1	R1	
Character 2	b12	b22	bn2		
		R2			
					1 0 1 1 0 1 1 1
					1 1 0 1 0 1 1 1
					0 0 1 1 1 0 1 0
					1 1 1 1 0 0 0 0 1
0 0 0 1 0 1 1					LRC
				Rm	
Character m	b1m	b2m			
Parity check character		bnm		cn+	0 1 0 1 1 1 1 1 ←
			c	1	
	c1	c2			
		bnm			

CRC Method

1. The frame is expressed in the form of a Polynomial F(x).0 1 1 1 1 1 1 0
2. Both the sender and receiver will agree upon a generator polynomial G(x) in advance.
3. Let _r‘be the degree of G(x).Append _r‘zero bits to the lower – order end offrame now it contains m+r bits.
4. Divide the bit string by G(x) using Mod 2 operation.
5. Transmitted frame [T(x)] = frame + remainder.
6. Divide T(x) by G(x) at the receiver end. If the result is a zero, then the frame is transmitted correctly.

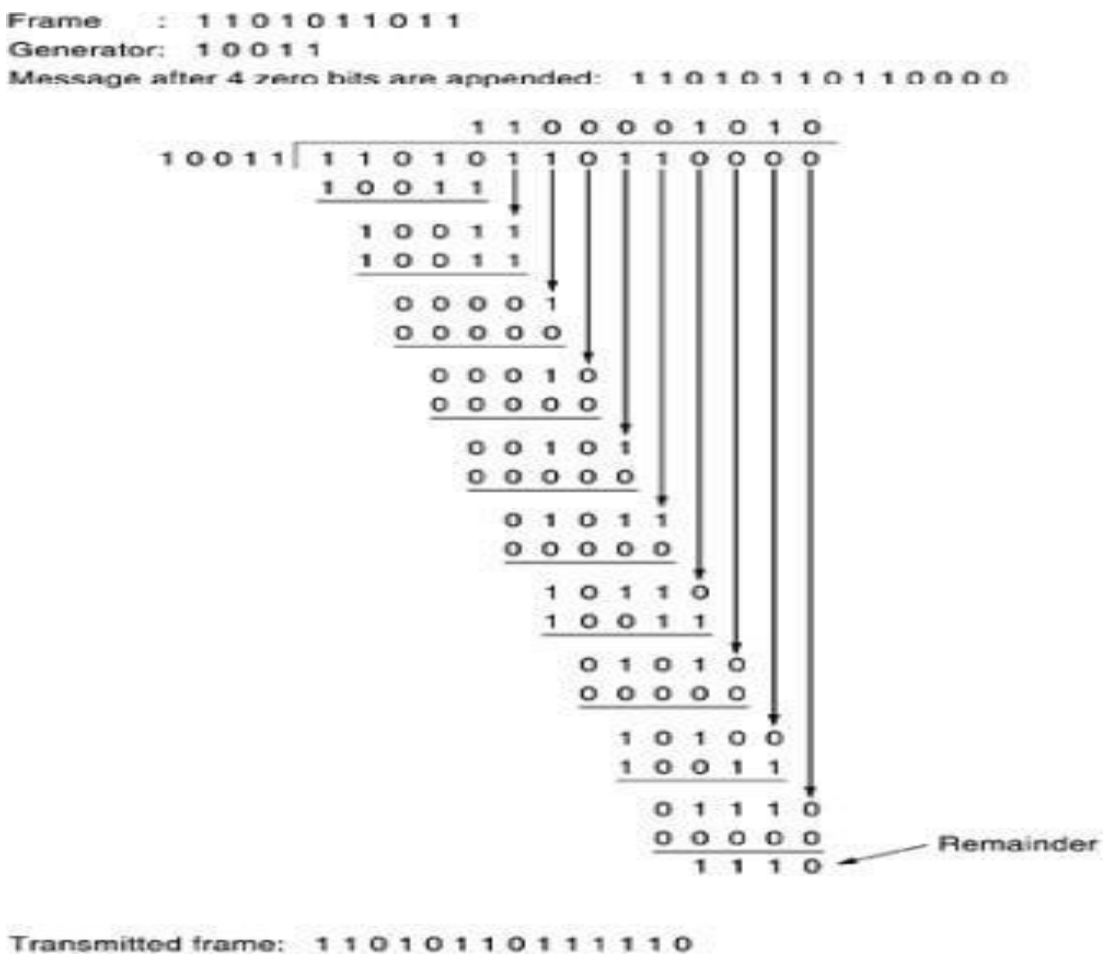


Figure 2-3. Calculation of the polynomial code checksum.

HAMMING CODES

Hamming codes provide another method for error correction. Error bits, called Hamming bits, are inserted into message bits at random locations. It is believed that the randomness of their locations reduces the odds that these Hamming bits themselves would be in error. This is based on a mathematical assumption that because there are so many more message bits compared with Hamming bits, there is a greater chance for a message bit to be in error than for a Hamming bit to be wrong. Determining the placement and binary value of the Hamming bits can be implemented using hardware,but it is often more practical to implement them using software. The number of bitsin a message (M) are counted and used to solve the following equation to determine the number of Hamming bits (H) to be used:

$2^H \geq M + H + 1$

Once the number of Hamming bits is determined, the actual placement of the bits into the message is performed. It is important to note that despite the random nature of the Hamming bit placements, the exact sample placements must be known and used by both the transmitter and receiver. Once the Hamming bits are inserted into their positions, the numerical values of the bit positions of the logic 1 bits in the original message are listed. The equivalent binary numbers of these values are added in the same manner as used in previous error methods by discarding all carry results. The sum produced is used as the states of the Hamming bits in the message. The numerical difference between the Hamming values transmitted and that produced at the receiver indicates the bit position that contains a bad bit, which is then inverted to correct it.

Ex. The given data

10010001100101(14- bits)

The number of hamming codes2^H

$\geq M + H + 1$

H = ? M = 14 to satisfy this equation H should be 5 i.e. 5 hamming code bits should be incorporated in the data bits.

1 0 0 1 0 0 0 1 1 0 H 0 H 1 H 0 H 1 H

Now count the positions where binary 1‘s are present. Add using mod 2 operation (Ex-OR). The result will give the Hamming code at the transmitter end.

1‘s position	Binary equivalent

2	-	0	0	0	1	0
6	-	0	0	1	1	0
11	-	0	1	0	1	1
12	-	0	1	1	0	0
16	-	1	0	0	0	0
19	-	1	0	0	1	1
Hamming code =		0	0	0	0	0

This Hamming code will be incorporated at the places of _H' in the data bits and the datawill be transmitted.

How to find out there is an error in the data?

Let the receiver received the 12th bit as zero. The receiver also finds out the Hammingcode in the same way as transmitter.

1's position	Binary equivalent				
2	0	0	0	1	0
6	0	0	1	1	0
11	0	1	0	1	1
16	1	0	0	0	0
19	1	0	0	1	1
Hamming code at the receiver	0	1	1	0	0

Hamming code at the Tx	0	0	0	0	0
Hamming code at the Rx	0	1	1	0	0
	0	1	1	0	0

The decimal equivalent for the binary is **12** so error is occurred at 12th place.

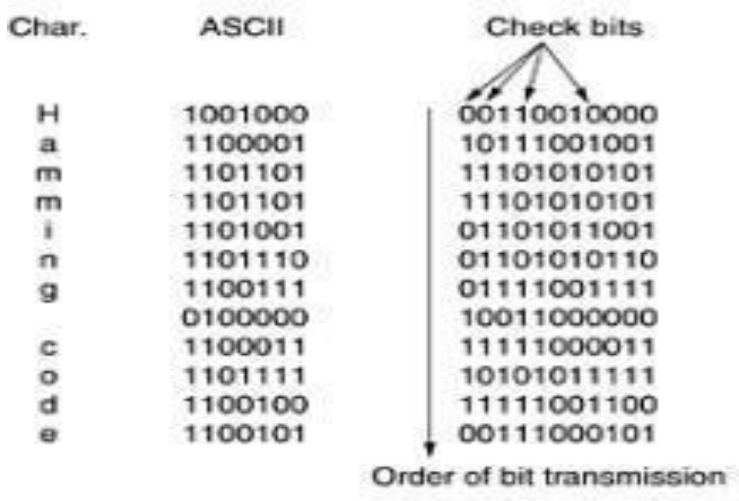


Figure 2-4. Use of a Hamming code to correct burst errors.

Hamming codes can only correct single errors. However, there is a trick that can be used to permit Hamming codes to correct burst errors. Sequences of k consecutive code words are arranged as a matrix, one codeword per row. Normally, the data would be transmitted one code word at a time, from left to right. To correct burst errors, the data should be transmitted one column at a time, starting with the leftmost column. When all k bits have been sent, the second column is sent, and so on, as indicated in Fig. 3-7. When the frame arrives at the receiver, the matrix is reconstructed, one column at a time. If a burst error of length k occurs, at most 1 bit in each of the k code words will have been affected, but the Hamming code can correct one error per codeword, so the entire block can be restored. This method uses kr check bits to make blocks of km data bits immune to a single burst error of length k or less.

Data Link Protocols

1. Unrestricted Simplex Protocol:

Data are transmitted in one direction only. Both the transmitting and receiving network layers are always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This thoroughly unrealistic protocol, which we will nickname "utopia," is shown in Fig. 3-10.

This is an unrealistic protocol, which has a nickname —Utopia‖.

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver.  The communication channel is assumed to be error free
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* send it on its way */
  }                                     /* Tomorrow, and tomorrow, and tomorrow,
                                           Creeps in this petty pace from day to day
                                           To the last syllable of recorded time.
                                              - Macbeth, V, v */
}


void receiver1(void)
{
  frame r;
  event_type event;                     /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
```

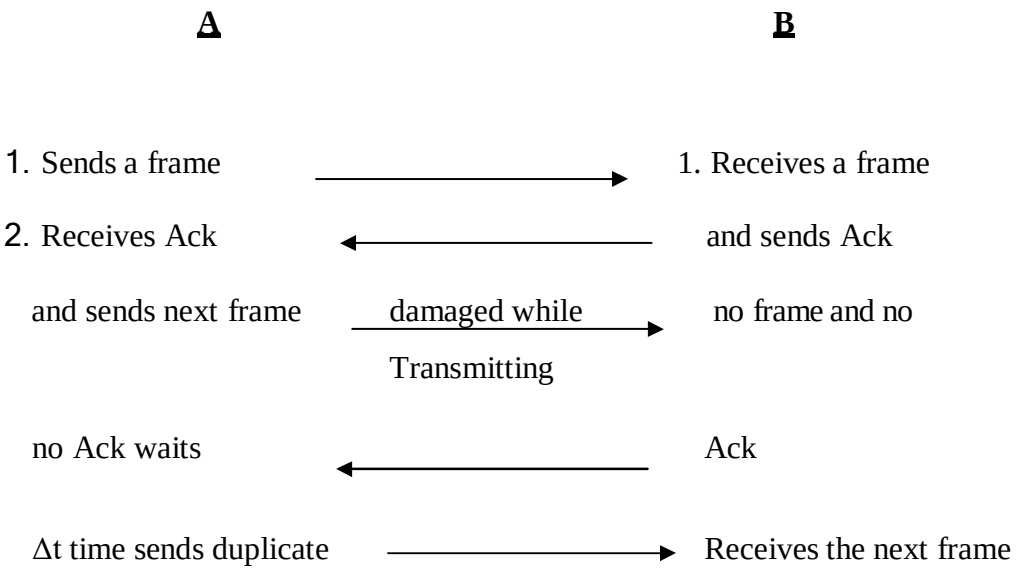
Figure 2-5. An unrestricted simplex protocol

2. **A simplex stop and wait protocol:**

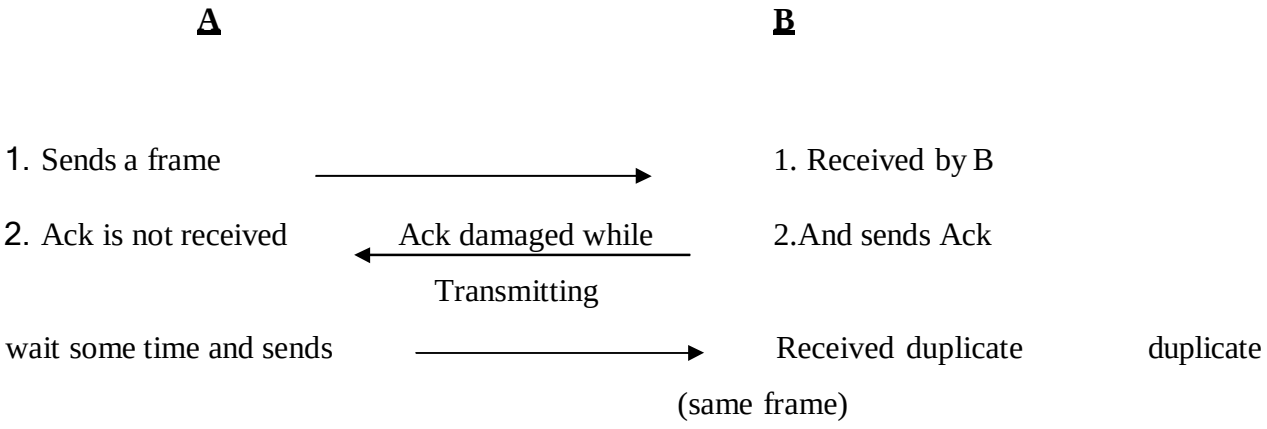
```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be erro
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;                     /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* bye-bye little frame */
      wait_for_event(&event);           /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers for frames */
  event_type event;                     /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
      to_network_layer(&r.info);        /* pass the data to the network layer */
      to_physical_layer(&s);            /* send a dummy frame to awaken sender */
  }
}
```

Figure 2-6. A simplex stop-and-wait protocol.

1. A simplex protocol for a noisy channel



When this protocol fails?

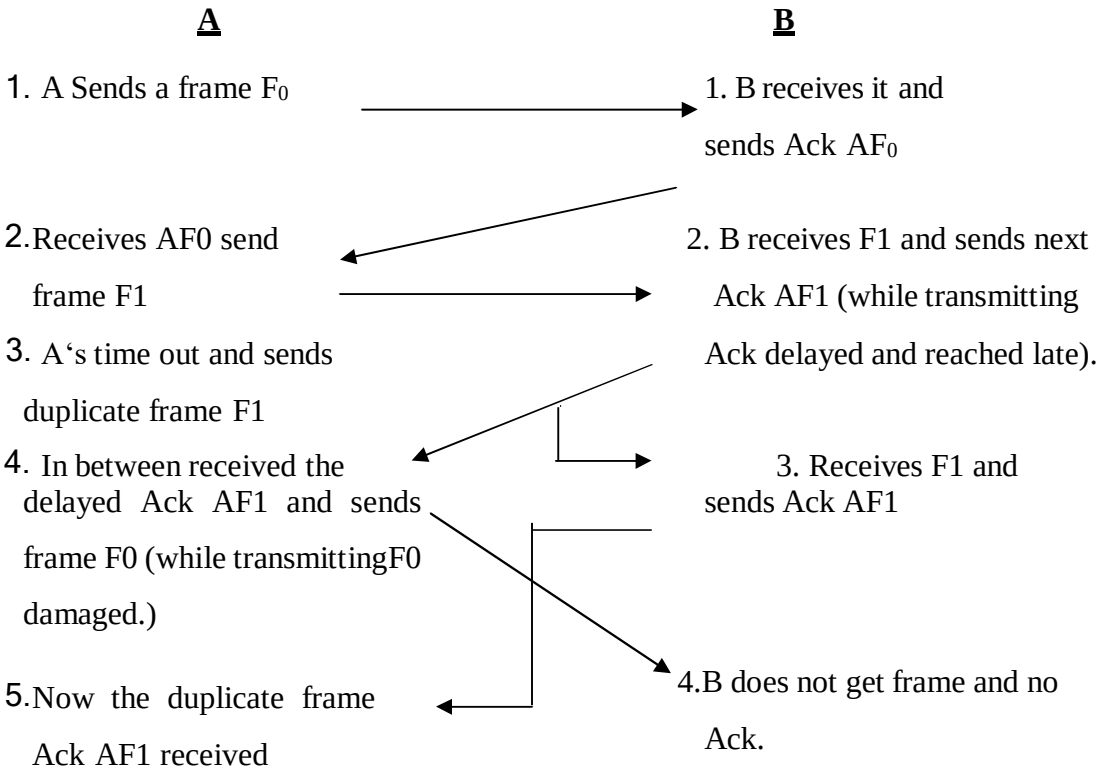


At this situation protocol fails because the receiver receives a duplicate frame and there is no way to find out whether the receiver frame is original or duplicate. So the protocol fails at this situation.

Now what is needed is some way for the Rx to distinguish a frame and a duplicate. To achieve this, the sender has to put a sequence number in the header of each frame it sends. The Rx can check the sequence number of each arriving frame to see if it is a new frame or a duplicate.

Here a question arises: What is the minimum number of bits needed for the sequence number? The ambiguity is between a frame and its successor. A 1-bit sequence number (0 or 1) is therefore sufficient. At each instant of time, the receiver excepts a particular sequence number next. Any arriving frame containing wrong sequence number is rejected as a duplicate. When a frame containing the correct sequence number arrives, it is accepted, passed to the networklayer and then expected sequence number is incremented i.e. 0 becomes 1 and one becomes 0. Protocols in which a sender waits for a positive ack before advancing to the next data item are often called PAR (positive ack with retransmission) or ARQ (automatic repeat request).

When this protocol fails?



6. Now A thinks that the Ack received is the ack of new frame F0 and A sends next frameF1. So a frame F0 is missed. At this situation this protocol fails.

SLIDING WINDOW PROTOCOLS

- Sliding window protocol allows the sender to send multiple frames before needing the acknowledgements.
- It is more efficient.

PIGGY BACKING

In most practical situations there is a need of transmitting data in both directions. This can be achieved by full duplex transmission. If this is done we have two separate physical circuits each with a _forward _ and _reverse‘ channel. In both cases, the reverse channel is almost wasted. To overcome this problem a technique called **piggy backing** is used.

The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as **piggy backing**.

However, piggybacking introduces a complication not present with separate acknowledgements. How long should the data link layer wait longer than the sender‘s timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements. Of course, the data link layer cannot foretell the future, so it must resort to some ad hoc scheme, such as waiting a fixed number of milli seconds. If a new packet arrives quickly, the acknowledgement is piggy backed onto it; otherwise, if no new packet has arrived by the end of this time period, the data link layer just sends a separate acknowledgement frame.

SLIDING WINDOW PROTOCOLS

In all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually $2^n - 1$ so the sequence number fits nicely in an n- bit field. The stop-and-wait sliding window protocol uses n=1, restricting the sequence numbers to 0 and 1, but more sophisticated versions can use arbitrary n.

The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The sender‘s window and the receiver‘swindow need not have the same lower and upper limits, or even have the same size. In some protocols they are fixed in size, but in others they can grow or shrink as frames are sent and received. The sequence numbers with in the sender‘s window represent frames sent but as yet not acknowledged. Whenever a new packet arrives from the network layer, it is given the next highest sequence number, and the upper edge of the window is advanced by one. When an acknowledgement comes in, the lower edge is advanced by one. In this way the continuously maintains a list of unacknowledged frames.

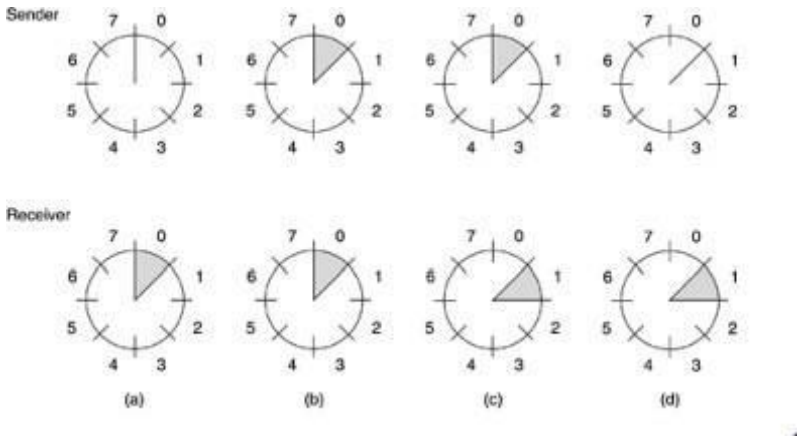


Figure 2-7. A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

A One-Bit Sliding Window Protocol

Before tackling the general case, let us first examine a sliding window protocol with a maximum window size of 1. Such a protocol uses stop-and-wait since the sender transmits a frame and waits for its acknowledgement before sending the next one.

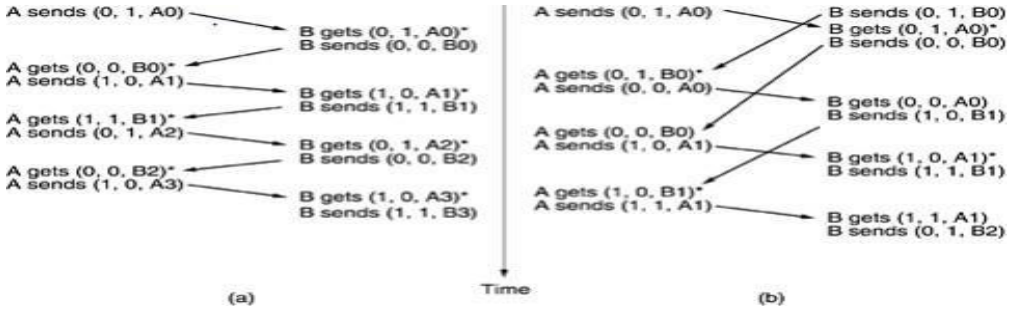


Figure 2-8. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

A Protocol Using Go Back N

PIPELINING

1. Up to now we made the assumption that the transmission time required for a frame to arrive atthe receiver plus the transmission time for the ack to come back is negligible.
2. Sometimes this is not true, when there is a long round trip propagation time is there.
3. In these cases round trip propagation time can have important implications for the efficiency of the bandwidth utilization.

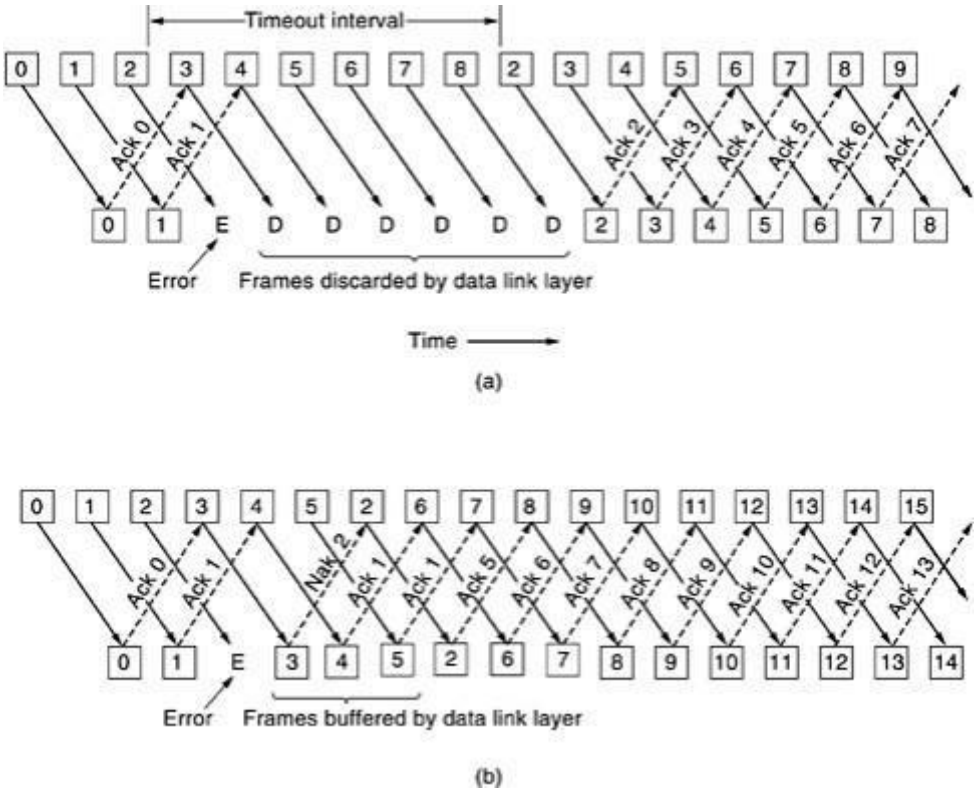


Figure 2-9. Pipelining and error recovery. Effect of an error when (a) receiver's window size is 1 and (b) receiver's window size is large.

Consider the below example.

Let the channel capacity b = 50Kbps.

- round trip propagation delay
  = 500
- Frame size = 1000bits
- Without considering the round trip propagation delay
- For one frame the time taken will be = 1000/500 ms= 20 ms

Considering the round trip propagation delay

A B

250 ms + 20 ms

250 ms

For one frame the time taken will be = 500 ms +20 ms

= 270 ms

The channel utilization = (20/520)*100 = 4%

i.e. We are wasting 96% of channel time. To overcome this problem we will go for a technique called **PIPELIING.**

In this technique, the sender is allowed to transmit upto _w _ frames before blocking, instead of just 1.With an appropriate choice of w the sender will be able to continuously transmit frames for a time equal to the round trip transmit time without filling up the window.

In the above example w would be at least 26 frames. (520/20 = 26 frames).By the time it has finished sending 26 frames, at t=520 ms, the ack for frame 0 will have just arrived. Thereafter ack will arrive every 20 ms, so the sender always gets permission to continue just when it needs it. Hence, we can say the sender window size is 26.

Derivation:

Let the channel capacity =	b Bps
Let the frame size =	l bit
Let the round trip delay =	R secs
To send one frame the time will be =	l/b secs
Due to round trip delay the time taken will be (l/b + R) Sec =	l+Rb/b Sec
The channel utilization is l/b (l/b + R) Sec =	(l / l + Rb) Sec
If l > bR the efficiency will be greater than 50%.	-
If l < bR the efficiency will be less than 50%	-
If l = bR the efficiency will be 50%	-

Pipelining frames over an unreliable channel raises some serious issues.

First, what happens if a frame in the middle of a long stream is damaged or lost? When a damaged frame arrives at the receiver, it obviously should discard, but whatshould the receiver do with all the correct frames following it?

There are two basic approaches to dealing with errors.

1. Go Back N
2. Selective repeat or Selective Reject

One way called in **go back n**, the receiver simply to discard all subsequent frames, sending no acknowledgements for the discard frames. In the other words, the data link layer refuses to accept any frame except the next one it must give to the network layer.

In Go back N, sender window size is N and receiver window size is always 1.

Go back N uses cumulative acknowledgements.

Go Back N

- Receiver maintains an acknowledgement timer.
- Each time the receiver receives a new frame, it starts a new acknowledgement timer.
- After the timer expires, receiver sends the cumulative acknowledgement for all the frames that are unacknowledged at that moment.

Stop and Wait ARQ	Go back N	Selective Repeat	Remarks	Stop and Wait ARQ
Efficiency	1 / (1+2a)	N / (1+2a)	N / (1+2a)	Go back N and Selective Repeat gives better efficiency than Stop and Wait ARQ.
Window Size	Sender Window Size = 1 Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = N	Buffer requirement in Selective Repeat is very large. If the system does not have lots of memory, then it is better to choose Go back N.
Minimum number of sequence numbers required	2	N+1	2 x N	Selective Repeat requires large number of bits in sequence number field.
Retransmissions required if a packet is lost	Only the lost packet is retransmitted	The entire window is retransmitted	Only the lost packet is retransmitted	Selective Repeat is far better than Go back N in terms of retransmissions required.
width Requirement	Bandwidth requirement is Low	Bandwidth requirement is high because even if a single packet is lost, entire window has to be retransmitted. Thus, if error rate is high, it wastes a lot of bandwidth.	Bandwidth requirement is moderate	Selective Repeat is better than Go back N in terms of bandwidth requirement.
CPU usage	Low	Moderate	High due to searching and sorting required at sender and receiver side	Go back N is better than Selective Repeat in terms of CPU usage.
Level of difficulty in Implementation	Low	Moderate	Complex as it requires extra logic and sorting and searching	Go back N is better than Selective Repeat in terms of implementation difficulty.
Acknowledgements	Uses independent acknowledgement for each packet	Uses cumulative acknowledgements (but may use independent acknowledgements as well)	Uses independent acknowledgement for each packet	Sending cumulative acknowledgements reduces the traffic in the network but if it is lost, then the ACKs for all the corresponding packets are lost.
Type of Transmission	Half duplex	Full duplex	Full duplex	Go back N and Selective Repeat are better in terms of channel usage.

- A new acknowledgement timer does not start after the expiry of old acknowledgement timer.
- It starts after a new frame is received.
- Consider after the expiry of acknowledgement timer, there is only one frame left to be acknowledged.
- Then, Go back N sends the independent acknowledgement for that frame.
- Go back N does not accept the corrupted frames and silently discards them.
- Go back N leads to retransmission of entire window if for any frame, no ACK is received by the sender.
- Efficiency = Sender Window Size in Protocol / (1 + 2a)
- Efficiency of Go back N = N / (1 + 2a)

Selective repeat or Selective Reject

- In SR protocol, sender window size is always same as receiver window size.
- SR protocol uses independent acknowledgements only.
- Receiver handles the situation efficiently by sending a negative acknowledgement (NACK).
- Negative acknowledgement allows early retransmission of the corrupted frame.
- It also avoids waiting for the time out timer to expire at the sender side to retransmit the frame.
- Receiver does not reject the out of order frames.
- Receiver accepts the out of order frames and sort them later.
- Thus, only the missing frame has to be sent by the sender.
- For sending the missing frame, sender performs searching and finds the missing frame.
- Then, sender selectively repeats that frame.
- Thus, only the selected frame is repeated and not the entire window.
- That is why, the protocol has been named as "**Selective Repeat Protocol**".
- Efficiency = Sender Window Size in Protocol / (1 + 2a)
- Efficiency of SR Protocol = N / (1 + 2a)

MEDIUM ACCESS CONTROL SUBLAYER (MAC)

Networks can be categories in to two ways

a) Point to point b) Broad cast channel

- In broadcast network, the key issue is how to share the channel amongseveral users.

- Ex *a conference call with five people*

-Broadcast channels are also called as multi-access channels or random accesschannels.

-Multi-access channel belong to a sublayer at the DL layer called the MAC sublayer.

The Channel Allocation problem:

a) **Static channel allocation** in LANs & MANs

i) FDM ii) **TDM**

Drawbacks: -1) Channel is wasted if one or more stations do not send data.

2) If users increases this will not support.

b) Dynamic channel allocation

i) Pure **ALOHA** & Slotted **ALOHA**

ii) CSMA — CSMA/CD

CSMA/CA

Pure ALOHA

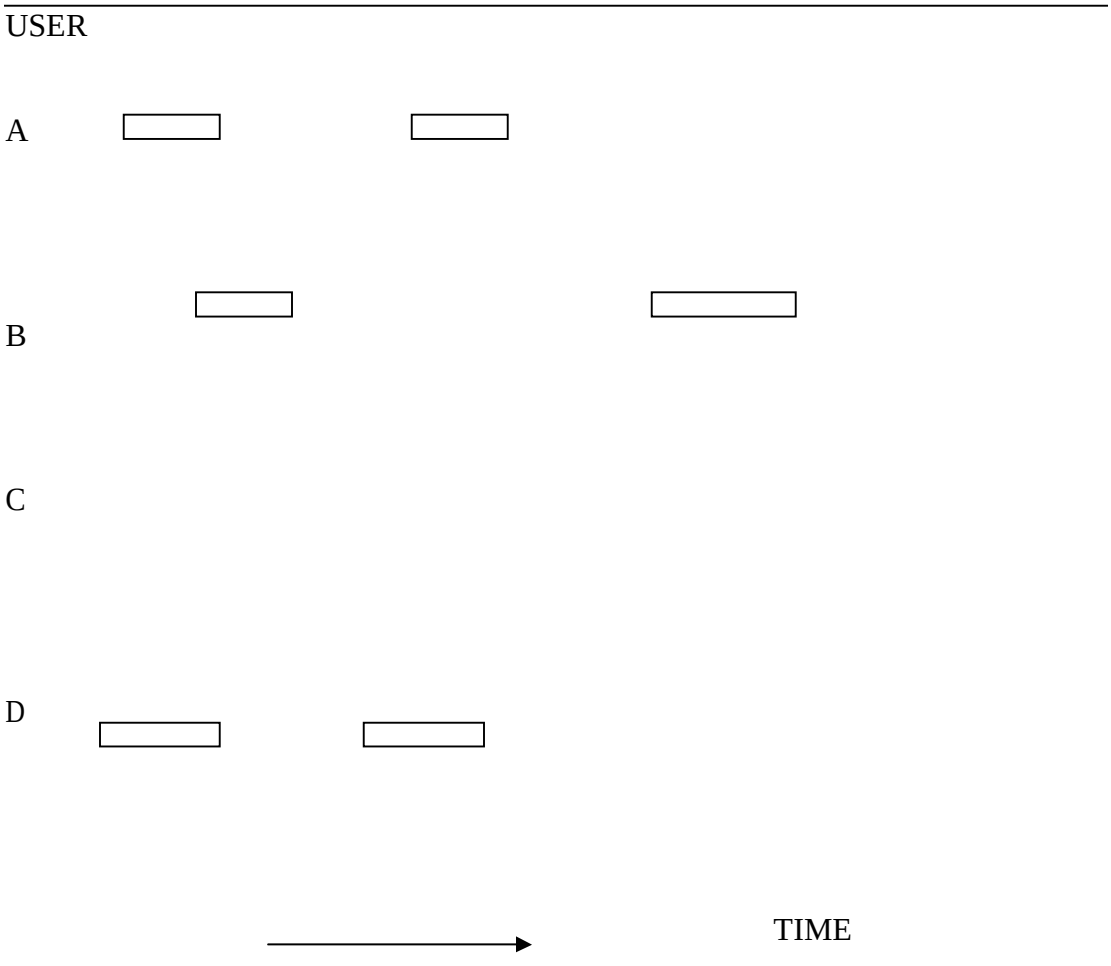
-1970‘s Norman Abramson end his colleagues devised this method, used ground –basedradio broad casting. This is called the **ALOHA** system.

-The basic idea, many users are competing for the use of a single shared channel.
-There are two versions of ALOHA: **Pure and Slotted**.
-Pure ALOHA does not require global time synchronization, where as in slotted ALOHAthe time is divided into discrete slots into which all frames must fit.

-Let users transmit whenever they have data to be sent.
-There will be collisions and all collided frames will be damaged.
-Senders will know through feedback property whether the frame is destroyed or not bylistening channel.[-With a LAN it is immediate, with a satellite, it will take 270m sec.]

-If the frame was destroyed, the sender waits random amount of time and again sendsthe frame.

-The waiting time must be random otherwise the same frame will collide over and over.



Frames are transmitted at completely arbitrary times

-Whenever two frames try to occupy the channel at the same time, there will be a collisionand both will be destroyed.

-We have to find out what is the efficiency of an ALOHA channel?

-Let us consider an infinite collection of interactive users sitting at their systems (stations).

-A user will always in two states **typing or waiting**.

-Let the _Frame time' denotes the time required to transmit one fixed length frame.

-Assume that infinite populations of users are generating new frames according to Poisson's distribution with mean N frames per frame time.

-If N>1 users are generating frames at a higher rate than the channel can handle.

-For reasonable throughput 0<N<1.

-In addition to new frames, the station also generates retransmission of frames.

-Old and new frames are G per frame time.

-G≥ N

-At low load there will be few collisions, so G ~ N

-Under all loads, the throughput $S = GP_o$, where P_o is the probability that a frame does notsuffer a collision.

-A frame will not suffer a collision if no other frames are sent with one frame time of itsstart.

-Let _t' be the time required to send a frame.

-If any other user has generated a frame between time t_o and t_o+t, the end of that framewill collide with the beginning of the shaded frame.

-Similarly, any other frame started b/w t_o+t and t_o+2t will bump into the end of the shadedframe.

-The probability that _k' frames are generated during a given frame time is given by thepossion distribution:

$$P_r[k] = \frac{G^k e^{-G}}{k!}$$

-The probability of zero frames is just e^{-G}

-In an interval two frame times long, the mean number at frames generated is 2G.

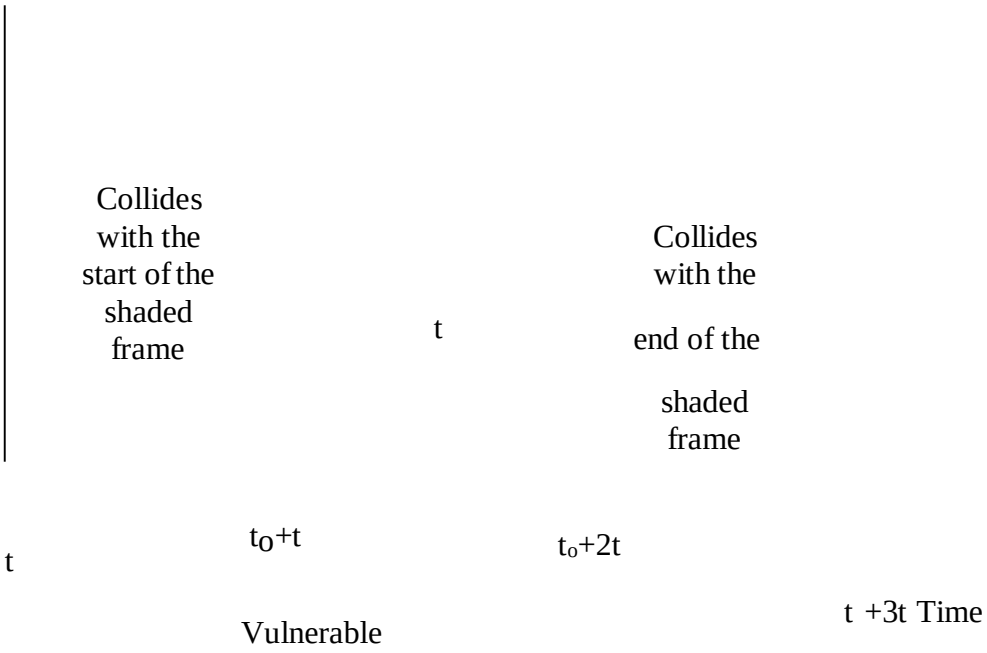
-The probability at no other traffic being initiated during the entire vulnerable period isgiven by

$$P_O = e^{-2G}$$

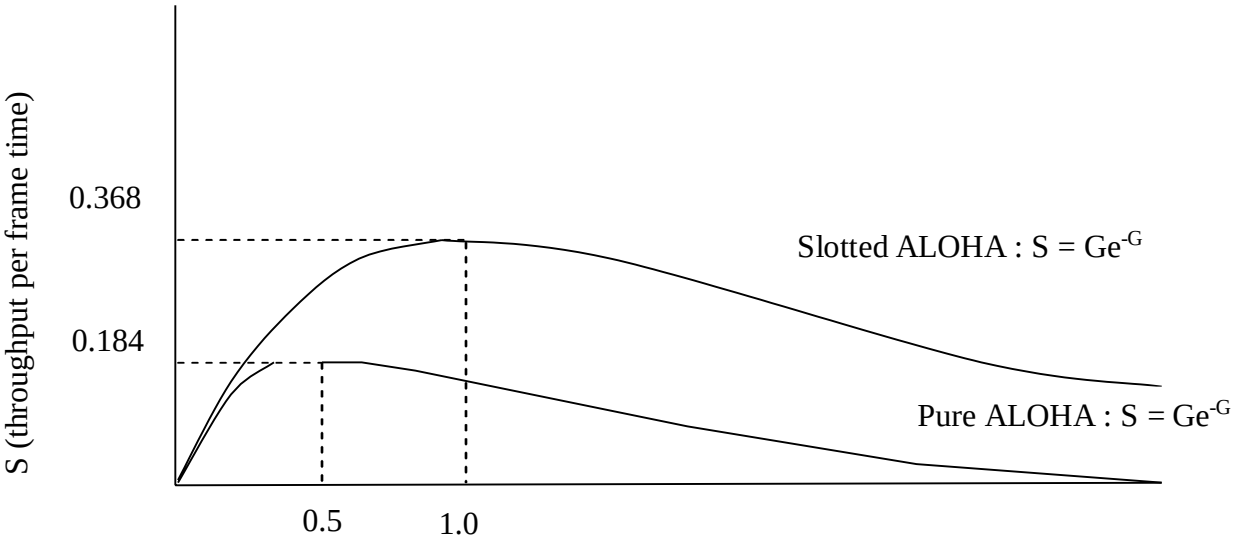
$$S = Ge^{-2G} \qquad [S=GP_o]$$

The Maximum through put occurs at G=0.5 with S=1/2e = 0.184

The channel utilization at pure ALOHA =18%.



Vulnerable period for the shaded frame



Throughput versus offered traffic for ALOHA systems

Slotted ALOHA

-In 1972, Roberts' devised a method for doubling the capacity of ALOHA system.

-In this system the time is divided into discrete intervals, each interval corresponding toone frame.

-One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

-In Roberts‘ method, which has come to be known as slotted ALOHA, in contrast to Abramson‘s pure ALOHA; a computer is not permitted to send whenever a carriage returnis typed.

-Instead, it is required to wait for the beginning of the next slot.

-Thus the continuous pure ALOHA is turned into a discrete one.

-Since the vulnerable period is now halved, the of no other traffic during the same slot as our test frame is e^{-G} which leads to

$S = Ge^{-G}$

- At G=1, slotted ALOHA will have maximum throughput.
- So S=1/e or about 0.368, twice that of pure ALOHA.
- The channel utilization is 37% in slotted ALOHA.

Carrier Sense Multiple Access Protocols

Protocols in which stations listen for a carrier (transmission) and act accordingly are called carries sense protocols.

Persistent CSMA

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 become 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 also 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. The longer the propagation delay the worse the performance of the protocol.

Even if the propagation delay is zero, there will be collisions. If two stations listen the channel, that is idle at the same, both will send frame and there will be collision.

Non persistent CSMA

In this, before sending, a station senses the channel. If no one else is sending, the station begins doing so itself. However, if the channel is busy, the station does not continually sense it but it waits a random amount of time and repeats the process.

This algorithms leads to better channel utilization but longer delays then 1- persistent CSMA.

With persistent CSMA, what happens if two stations become active when a third station is busy? Both wait for the active station to finish, and then simultaneously launch a packet, resulting a collision. There are two ways to handle this problem.

a) P-persistent CSMA b) exponential backoff.

P-persistent CSMA

The first technique is for a waiting station not to launch a packet immediately when the channel becomes idle, but first toss a coin, and send a packet only if the coin comes up heads. If the coin comes up tails, the station waits for some time (one slot for slotted CSMA), then repeats the process. The idea is that if two stations are both waiting for the medium, this reduces the chance of a collision from 100% to 25%. A simple generalization of the scheme is to use a biased coin, so that the probability of sending a packet when the medium becomes idle is not 0.5, but p, where $0 < p < 1$. We call such a scheme **P-persistent CSMA**. The original scheme, where p=1, is thus called 1-persitent CSMA.

Exponential backoff

The key idea is that each station, after transmitting a packet, checks whether the packet transmission was successful. Successful transmission is indicated either by an explicit acknowledgement from the receiver or the absence of a signal from a collision detection circuit. If the transmission is successful, the station is done. Otherwise, the station retransmits the packet, simultaneously realizing that at least one other station is also contending for the medium. To prevent its retransmission from colliding with the other station‘s retransmission, each station backs off (that is, idles) for a random time chosen from the interval

[0,2*max-propagation_delay] before retransmitting its packet. If the retransmission also fails, then the station backs off for a random time in the interval [0,4* max_propagation_delay], and tries again. Each subsequent collision doubles the backoff interval length, until the retransmission finally succeeds. On a successful transmission,the backoff interval is reset to the initial value. We call this type of backoff exponential backoff.

CSMA/CA

In many wireless LANS, unlike wired LANS, the station has no idea whether the packet collided with another packet or not until it receives an acknowledgement from receiver. In this situation, collisions have a greater effect on performance than with CSMA/CD, where colliding packets can be quickly detected and aborted. Thus, it makes sense to try to avoid collisions, if possible. CSMA/CA is basically p-persistence, with the twist that when the medium becomes idle, a station must wait for a time called the interframe spacing or IFS before contending for a slot. A station gets a higher priority if it is allocated smaller inter frame spacing.

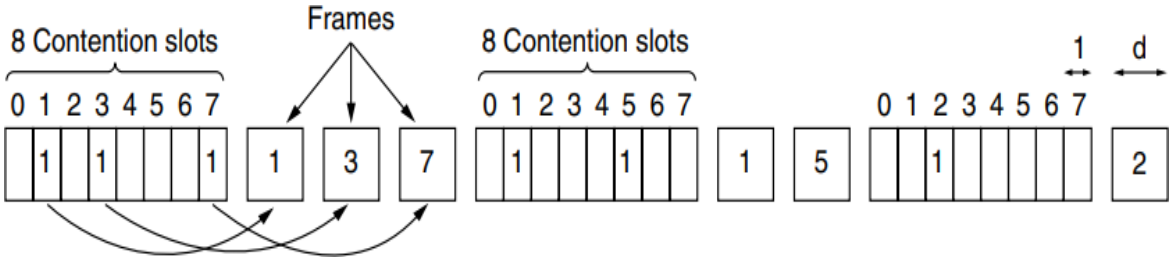
When a station wants to transmit data, it first checks if the medium is busy. If it is, it continuously senses the medium, waiting for it to become idle. When the medium becomes idle, the station first waits for an interframe spacing corresponding to its priority level, then sets a contention timer to a time interval randomly selected in the range[0,CW], where CW is a predefined contention window length. When this timer expires, it transmits a packet and waits for the receiver to send an ack. If no ack is received, the packet is assumed lost to collision, and the source tries again, choosing a contentiontimer at random from an interval twice as long as the one before(binary exponential backoff). If the station senses that another station has begun transmission while it was waiting for the expiration of the contention timer, it does not reset its timer, but merely freezer it, and restarts the countdown when the packet completes transmission. In this way, stations that happen to choose a longer timer value get higher priority in the next round of contention.

Collision-Free Protocols

A Bit-Map Protocol

In the basic bit-map method, each contention period consists of exactly N slots. If station0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the

opportunity to transmit a 1during slot 1, but only if it has a frame queued. In general, station j may announce the fact that it has a frame to send by inserting a 1 bit into slot j. after all N slots have passed by, each station has complete knowledge of which stations with to transmit.



The basic bit-map protocol

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another Nbit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols like this in which the desire to transmit is broadcast before the actual transmission are called reservation protocols.

Binary Countdown

A problem with the basic bit-map protocol is that the overhead is 1 bit per station. A station wanting to use the channel now broadcasts its address as a binary bit string, starting with the high- order bit. All addresses are assumed to be the same length. The bits in each address position from different stations are BOOLEAN ORed together. We will call this protocol binary countdown. It is used in Datakit.

To avoid conflicts, an arbitration rule must be applied: as soon as a station sees that a high-order bit position that is 0 in its address has been overwritten with a 1, it gives up. For example, if stations 0010, 0100, 1001, and 1010 are all trying to get the channel, in the first bit time the stations transmit 0, 0, 1, and 1, respectively. These are ORed together to form a 1. Stations 0010 and 0100 see the 1 and know that a higher-numbered station is competing for the channel, so they give up for the current round. Stations 1001 and 1010 continue. The next bit is 0, and both stations continue. The next bit is 1, so station 1001 gives up. The winner is station 1010 because it has the highest address. After winning the bidding, it may now transmit a frame, after which another bidding cycle starts. The protocol is illustrated in Fig. 4-8. It has the property that higher-numbered

stations have a higher priority than lower-numbered stations, which may be either good or bad, depending on the context.

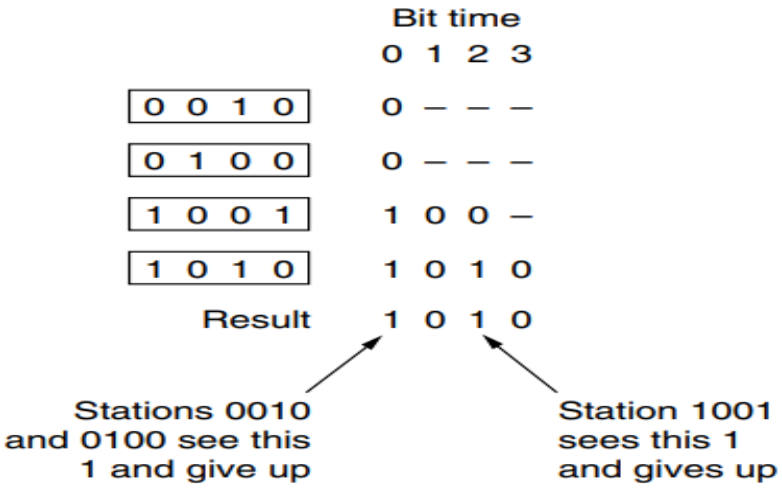


Figure 2-10. The binary countdown protocol. A dash indicates silence

Ethernet, probably the most ubiquitous kind of computer network in the world. Two kinds of Ethernet exist: classic Ethernet, which solves the multiple access problem using the techniques we have studied in this chapter; and switched Ethernet, in which devices called switches are used to connect different computers. It is important to note that, while they are both referred to as Ethernet, they are quite different. Classic Ethernet is the original form and ran at rates from 3 to 10 Mbps. Switched Ethernet is what Ethernet has become and runs at 100, 1000, and 10,000 Mbps, in forms called fast Ethernet, gigabit Ethernet, and 10 gigabit Ethernet.

Classic Ethernet Physical Layer

Classic Ethernet snaked around the building as a single long cable to which all the computers were attached. This architecture is shown in Fig. 4-13. The first variety, popularly called thick Ethernet, resembled a yellow garden hose, with markings every 2.5 meters to show where to attach computers. (The 802.3 standard did not actually require the cable to be yellow, but it did suggest it.) It was succeeded by thin Ethernet, which bent more easily and made connections using industry-standard BNC connectors. Thin Ethernet was much cheaper and easier to install, but it could run for only 185 meters per segment (instead of 500 m with thick Ethernet), each of which could handle only 30 machines (instead of 100). Each version of Ethernet has a maximum cable length per segment (i.e., unamplified length) over which the signal will propagate. To allow larger networks, multiple cables can be connectedby repeaters. A repeater is a physical layer device that receives, amplifies (i.e., regenerates), and retransmits signals in both directions. As far as the software is concerned, a series of cable segments connected by repeaters is no different from a single cable (except for a small amount of delay introducedby the repeaters).

Over each of these cables, information was sent using the Manchester encoding we studied in Sec. 2.5. An Ethernet could contain multiple cable segments and multiple repeaters, but no two transceivers could be more than 2.5 km apart and no path between any two transceivers could traverse more than four repeaters. The reason for this restriction was so that the MAC protocol, which we will look at next, would work correctly.

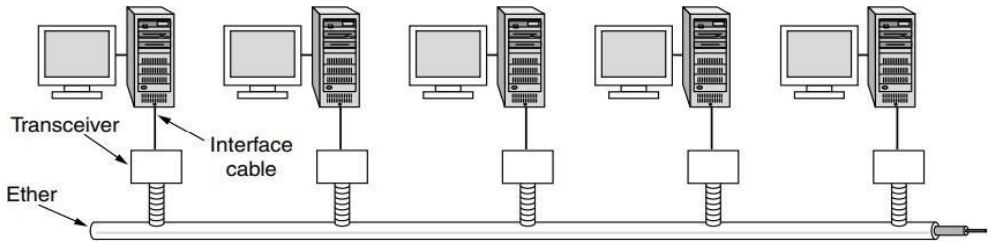


Figure 2-11. Architecture of classic Ethernet.

The 802.3 MAC sub layer protocol:

I) Preamble:
Each frame start with a preamble of 7 bytes each contains a bit pattern 10101010.

II) Start of frame byte:
It denotes the start of the frame itself. It contains 10101011.

III) Destination address:
This gives the destination address. The higher order bit is zero for ordinary address and 1for group address (Multi casting). All bits are 1s in the destination field frame will be delivered to all stations (Broad casting).

The 46^{th} bit (adjacent to the high-order bit) is used to distinguish local from global addresses.

IV) Length field:
This tells how many bytes are present in the data field from 0 to 1500.

V) Data field:
This contains the actual data that the frame contains.

VI) Pad:
Valid frame must have 64 bytes long from destination to checksum. If the frame size less than 64 bytes pad field is used to fill out the frame to the minimum size.

VII) Checksum:

It is used to find out the receiver frame is correct or not. CRC will be used here.

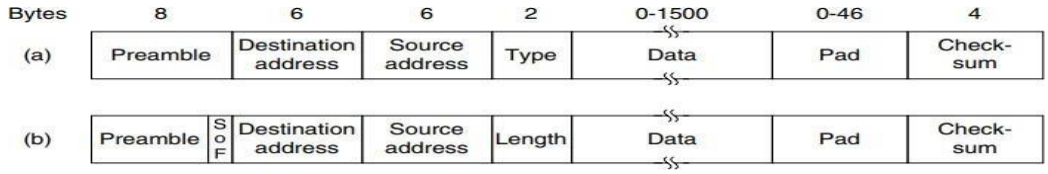


Figure 2-13. Frame formats. (a) Ethernet (DIX). (b) IEEE 802.3.

Switched Ethernet:

- 10 Base-T Ethernet is a shared media network.
- The entire media is involved in each transmission.

The HUB used in this network is a passive device. (not intelligent).

In switched Ethernet the HUB is replaced with switch. Which is a active device(intelligent) rest ofthe packet.

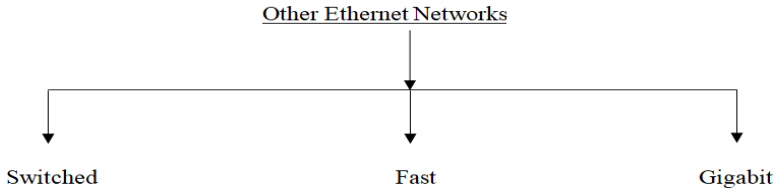
Source address The source address field is a four-byte (32-bit) Internet address. It identifies theoriginal source of the datagram.

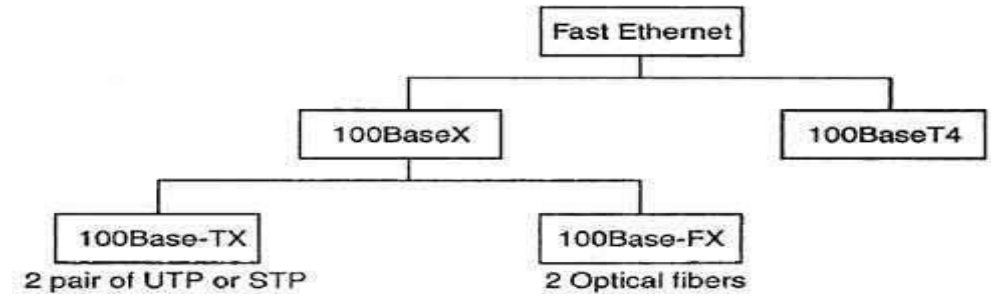
Destination address The destination address field is a four-byte (32-bit) Internet address. Itidentifies the final destination of the datagram.

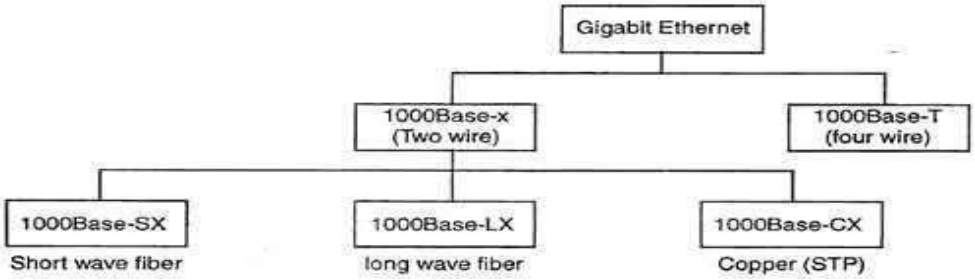
Options The options field gives more functionality to IP datagram. It can carry fields that controlrouting, timing, management, and alignment.

ADDRESSING

In addition to the physical address the internet requires an additional addressing convention : anaddress that identifies the connection of a host to its network.




Fast Ethernet
100BaseX
100BaseT4
100Base-TX
2 pair of UTP or STP
100Base-FX
2 Optical fibers

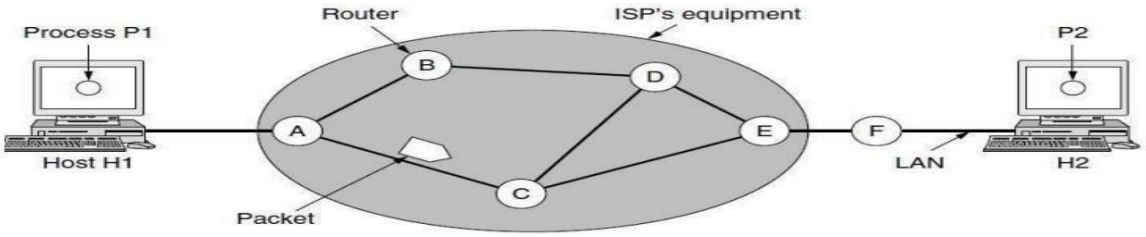

Gigabit Ethernet
1000Base-x
(Two wire)
1000Base-T
(four wire)
1000Base-SX
Short wave fiber
1000Base-LX
long wave fiber
1000Base-CX
Copper (STP)

UNIT III

NETWORK LAYER DESIGN ISSUES:

In the following sections, we will give an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the network.

STORE-AND-FORWARD PACKET SWITCHING: before starting to explain the details of the network layer, it is worth restating the context in which the network layer protocols operate. This context can be seen in. The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host H1 is directly connected to one of the ISP's routers, A, perhaps as a home computer that is plugged into a DSL modem. In contrast, H2 is on a LAN, which might be an office Ethernet, with a router, F, owned and operated by the customer. This router has a leased line to the ISP's equipment. We have shown F as being outside the oval because it does not belong to the ISP. For the purposes of this chapter, however, routers on customer premises are considered part of the ISP network because they run the same algorithms as the ISP's routers (and our main concern here is algorithms).



The environment of the network layer protocols.

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destinationhost, where it is delivered. This mechanism is store-and-forward packet switching.

SERVICES PROVIDED TO THE TRANSPORT LAYER:

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs. Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer. This freedom often degenerates into a raging battle between two warring factions. The discussion centers on whether the network layer should provide connection-oriented service or connectionless service.

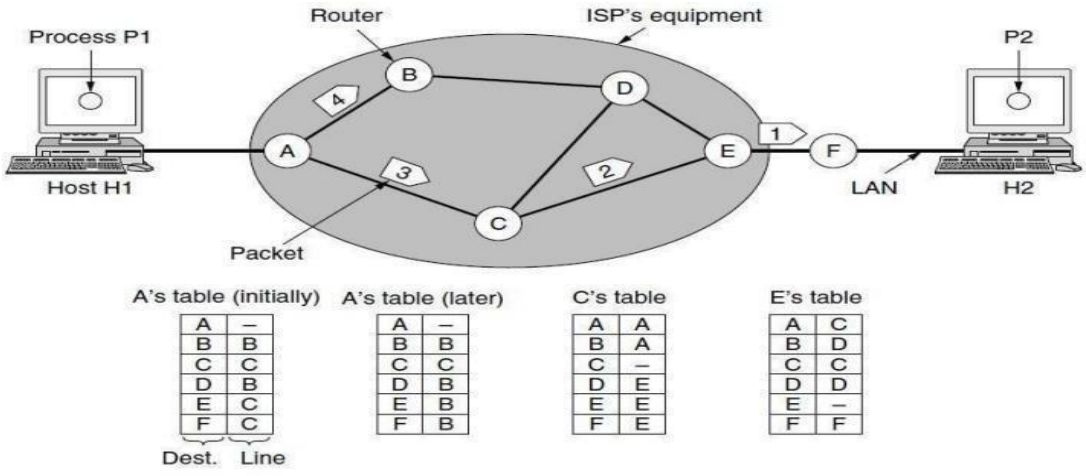
One camp (represented by the Internet community) argues that the routers' job is moving packets around and nothing else. In this view (based on 40 years of experience with a real computer network), the network is inherently unreliable, no matter how it is designed. Therefore, the hosts should accept this fact and do error control (i.e., error detection and correction) and flow control themselves. This viewpoint leads to the conclusion that the network service should be

connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway and there is usually little to be gained by doing it twice. This reasoning is an example of the end-to-end argument, a design principle that has been very influential in shaping the Internet (Saltzer et al., 1984). Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.

The other camp (represented by the telephone companies) argues that the network should provide a reliable, connection-oriented service. They claim that 100 years of successful experience with the worldwide telephone system is an excellent guide. In this view, quality of service is the dominant factor, and without connections in the network, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video. Even after several decades, this controversy is still very much alive. Early, widely used data networks, such as X.25 in the 1970s and its successor Frame Relay in the 1980s, were connection-oriented. However, since the days of the ARPANET and the early Internet, connectionless network layers have grown tremendously in popularity. The IP protocol is now an ever-present symbol of success. It was undeterred by a connection-oriented technology called ATM that was developed to overthrow it in the 1980s; instead, it is ATM that is now found in niche uses and IP that is taking over telephone networks. Under the covers, however, the Internet is evolving connection-oriented features as quality of service becomes more important. Two examples of connection-oriented technologies are MPLS (Multi Protocol Label Switching and VLANs, which we saw in. Both technologies are widely used.

IMPLEMENTATION OF CONNECTIONLESS SERVICE: Having looked at the two classes of service the network layer can provide to its users, it is time to see how this layer works inside. Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called datagrams (in analogy with telegrams) and the network is called a datagram network. If connection- oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit), in analogy with the physical circuits set up by the88 telephone system, and the network is called a virtual-circuit network

Let us now see how a datagram network works. Suppose that the process P1 in Fig. has a long message for P2. It hands the message to the transport layer, with instructions to deliver it to process P2 on host H2. The transport layer code runs on H1, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer, probably just another procedure within the operating system.



Routing within a datagram network.

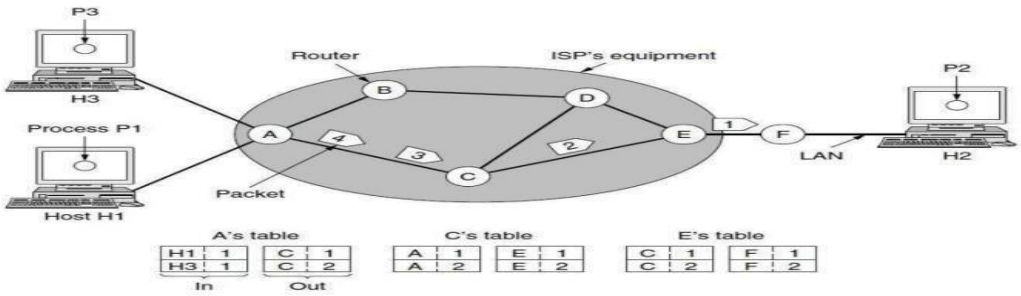
Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2,3, and 4, and send each of them in turn to router A using some point-to-point protocol, for example, PPP. At this point the ISP takes over. Every router has an internal table telling it where to send packets for each of the possible destinations.

Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly connected lines can be used. For example, in Fig., A has only two outgoing lines—to B and to C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. A's initial routing tableis shown in the figure under the label ''initially.'' At A, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded according to A's table, onto the outgoing link to C within a new frame. Packet 1 is then forwarded to E and then to F. When it gets to F, it is sent within a frame over the LAN to H2. Packets 2 and 3 follow the same route.

However, something different happens to packet 4. When it gets to A it is sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a 89 different route than that of the first three packets. Perhaps it has learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ''later.'' The algorithm that manages the tables and makes the routing decisions is called the routing algorithm. Routing algorithms are one of the main topics we will study in this chapter. There are several different kinds of them, as we will see. IP (Internet Protocol), which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and 128 bits in IPv6 packets.

IMPLEMENTATION OF CONNECTION-ORIENTED SERVICE:

For connection-oriented service, we need a virtual-circuit network. Let us see how that works. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection- oriented service, each packet carries an identifier telling which virtual circuit it belongs to. As an example, consider the situation shown in Fig. Here, host H1 has established connection 1 with host H2. This connection is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.



Now let us consider what happens if H3 also wants to establish a connection to H2. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the network to establish the virtual circuit. This leads to the second row in the tables. Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this. For this reason, A assignsa different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets.

In some contexts, this process is called label switching. An example of a connection-oriented network service is MPLS (Multi-Protocol Label Switching). It is used within ISP networks in the Internet, with IP packets wrapped in an MPLS header having a 20-bit connection identifier or label. MPLS is often hidden from customers, with the ISP establishing long-term connections for large amounts of traffic, but it is increasingly being used to help when quality of service is important but also with other ISP traffic management tasks.

Comparison of Virtual-Circuit and Datagram Networks Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in Fig, although purists could probably find a counterexample for everything in the figure.

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

Comparison of datagram and virtual-circuit networks.

Inside the network, several trade-offs exist between virtual circuits and data grams. One trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources. However, once this price is paid, figuring out what to do with a data packet in a virtual-circuit network is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram network, no setup is needed but a more complicated lookup procedure is required to locate the entry for the destination. A related issue is that the destination addresses used in datagram networks are longer than circuit numbers used in virtual-circuit networks because they have a global meaning. If the packets tend to be fairly short, including a full destination address in every packet may represent a significant amount of overhead and hence a waste of bandwidth. Yet another issue is the amountof table space required in router memory. A datagram network needs to have an entry for every possible destination, whereas a virtual-circuit network just needs an entry for each virtual circuit.

However, this advantage is somewhat illusory since connection setup packets have to be routed too, and they use destination addresses, the same as datagram's do. Virtual circuits have some advantages in guaranteeing quality of service and avoiding congestion within the network because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established. Once the packets start arriving, the necessary bandwidth and router capacity will be there. With a datagram network, congestion avoidance is more difficult. For transaction processing systems (e.g., stores calling up to verify credit card purchases), the overhead required to set up and clear a virtual circuit may easily dwarf the use of the circuit.

If the majority of the traffic is expected to be of this kind, the use of virtual circuits inside the network makes little sense. On the other hand, for long-running uses such as VPN traffic between two corporate offices, permanent virtual circuits (that are set up manually and last for months or years) may be useful. Virtual circuits also have a vulnerability problem. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted. In contrast, if a datagram router goes down, only those users whose packets were queued in the router at the time need suffer (and probably not even then since the sender is likely to retransmit them shortly). The loss of a communication line is fatal to virtual circuits using it, but can easily be compensated for if datagram's are used. Datagram's also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

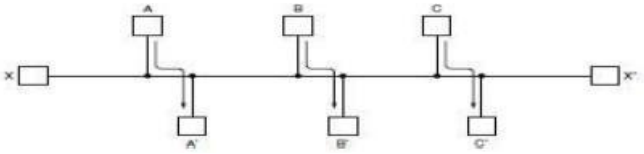
ROUTING ALGORITHMS: The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network segment. The algorithms that choose the routes and the data structures that they use are a major area of network layer design. The routing algorithm is that part of the network layer software responsible for deciding which output line anincoming packet should be transmitted on. If the network uses datagram's internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the already established route. The latter case is sometimes called session routing because 97 a route remains in force for an entire session (e.g., while logged in over a VPN). It is sometimes useful to make a distinction between routing, which

is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can thinkof a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line touse for it in the routing tables. This process is forwarding. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet sent or only when new connections are established, certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and efficiency. Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without system-wide failures. During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times.

The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted. Imagine the havoc if the network needed to be rebooted every time some router crashed! Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to a fixed set of paths, no matter how long they run. A stable algorithm reaches equilibrium and stays there. It should converge quickly too, since communication may be disrupted until the routing algorithm has reached equilibrium. Fairness and efficiency may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals. As a simple example of this conflict, look at Fig. Suppose that 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. Before we can even attempt to find trade-offs between fairness and efficiency, we must decide what it is we seek to optimize. Minimizing the mean packet delay is an obvious candidate to send traffic through the network effectively, but so is maximizing total network throughput. Furthermore, these two goals are also in conflict, since operating any queuing system near capacity implies a long queuing delay. As a compromise, many networks attempt to minimize the distance a packet must travel, or simply reduce the number of hops a packet must make. Either choice tends to improve the delay andalso reduce the amount of bandwidth consumed per packet, which tends to improve the overall network throughput as well.Routing algorithms can be grouped into two major classes: non adaptive and adaptive. Non adaptive algorithms do not base their routing decisions on any measurements or estimates of the current topology and traffic



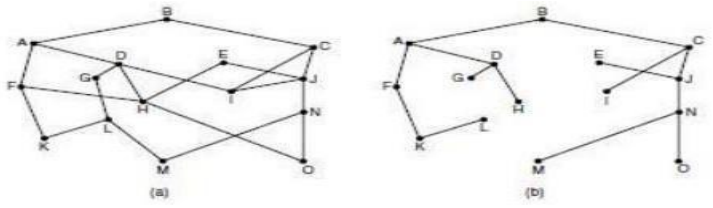
Network with a conflict between fairness and efficiency.

Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear. For example, router F in Fig. should send packets headed into the network to router E regardless of the ultimate destination. Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well.

These dynamic routing algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes(e.g., when the topology changes, or every T seconds as the load changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).In the following sections, we will discuss a variety of routing algorithms. The algorithms cover delivery models besides sending a packet from a source to a destination. Sometimes the goal is to send the packet to multiple, all, or one of a set of

destinations. All of the routing algorithms we describe here make decisions based on the topology; we defer the possibility of decisions based on the traffic levels to Sec.

The Optimality Principle: Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle (Bellman, 1957). It states that 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. To see this, call the part of the route from I to J r1 and the rest of the route r 2. If a route better than r 2 existed from J to K, it could be concatenated with r 1 to improve the route from I to K, contradicting our statement that r 1r 2 is optimal. As a direct consequence of the optimality principle, we can see that 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 and is illustrated in Fig. where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.

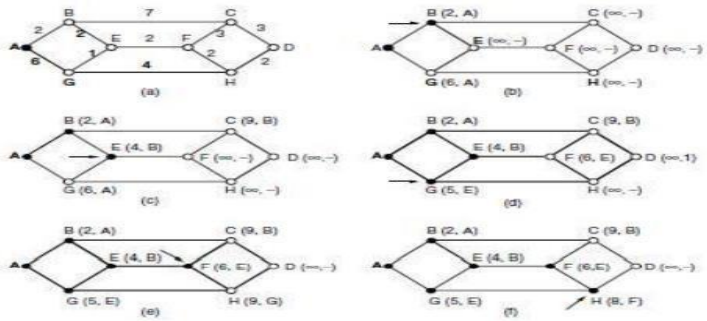


(a) A network. (b) A sink tree for router *B*.

Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a DAG (Directed Acyclic Graph). DAGs have no loops. We will use sink trees as convenient shorthand for both cases. Both cases also depend on the technical assumption that the paths do not interfere with each other so, for example, a traffic jam on one path will not cause another path to divert. Since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. In practice, life is not quite this easy. Links and routers can go down and come backup during operation, so different routers may have different ideas about the current topology. Also, we have quietly finessed the issue of whether each router has to individually acquire the information on which to base its sink tree computation or whether this information is collected by some other means. We will come back to these issues shortly. Nevertheless, the optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured.

Shortest Path Algorithm: Let us begin our study of routing algorithms with a simple technique for computing optimal paths given a complete picture of the network. These paths are the ones that we want a distributed routing algorithm to find, even though not all routers may know all of the details of the network. The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

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



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

However, many other metrics besides hops and physical distance are also possible. For example, each edge could be labeled with the mean delay of a standard test packet, as measured by hourly runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest edges or kilometers. In the general case, the labels on the edges could be computed as a function of the distance, bandwidth, average traffic, communication cost, 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. Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path. The distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter. To illustrate how the labeling algorithm works, look at the weighted, undirected graph of Fig., 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, eachof the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. If the network had more than one shortest path from A to D and we wanted to find all of them, we would need to remember all of the probe nodes that could reach a node with the same distance. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. (b). this one becomes the new working node.

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 relabeled. 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 labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure shows the first six steps of the algorithm. To see why the algorithm works, look at Fig. (c).At this point we have just made E permanent. Suppose that there were a shorter path than ABE, say AXYZE (for some X and Y). There are two possibilities: either node Z has already been made permanent, or it has not been. If it has, 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 labeled. If the label at Z is greater than or equal to that at E, then AXYZE cannot be a shorter path than ABE. If the label is less than thatof E, then Z and not E will become permanent first, allowing E to be probed from Z. This algorithm is given in Fig. 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., we compute the shortest path starting at the terminal node, t, rather than at the source node, s. Since the shortest paths from t to s in an undirected graph are the sameas the shortest paths from s to t, it does not matter at which end we begin. The reason for searching backward is that each node is labeled with its predecessor rather than its successor. When the final path is copied into the output variable, path, the path is thus reversed. The two reversal effects cancel, and the answer is produced in the correct order.

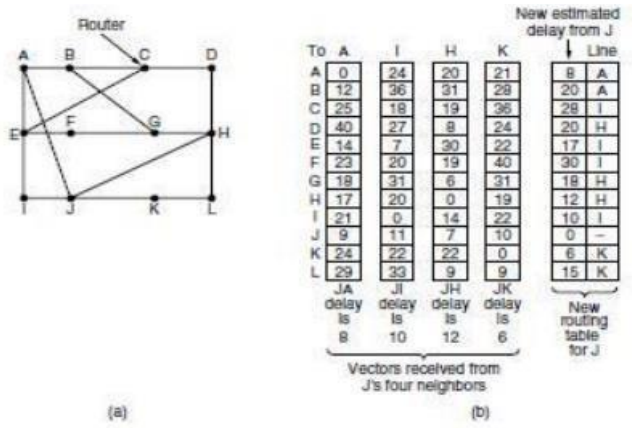
Flooding: when a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is flooding, in which every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding obviously 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 that 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 network.

Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before. A better technique for damming the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time. One way to achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

To prevent the list from growing without bound, each list should be augmented by a counter, k, meaning that all sequence numbers through k have been seen. When a packet comes in, it is easy to check if the packet has already been flooded (by comparing its sequence number to k; if so, it is discarded. Furthermore, the full list below k is not needed, since k effectively summarizes it. Flooding is not practical for sending most packets, but it does have some important uses. First, it ensures that a packet is delivered to every node in the network. This may be wasteful if there is a single destination that needs the packet, but it is effective for broadcasting information. In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property. Second, flooding is tremendously robust. Even if large numbers of routers are blown to bits (e.g., in a military network located in a war zone), flooding will find a path if one exists, to get a packet to its destination. Flooding also requires little in the way of setup. The routers only need to know their neighbors. This means that flooding can be used as a building block for other routing algorithms that are more efficient but need more in the way of setup. Flooding can also be used as a metric against which other routing algorithms can be compared. Flooding always chooses the shortest path because it chooses every possible path in parallel. Consequently, no other algorithm can producea shorter delay (if we ignore the overhead generated by the flooding process itself).

Distance Vector Routing: Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology. Two dynamic algorithms in particular, distance vector routing and link state routing, are the most popular. In this section, we will look at the former algorithm. In the following section, we will study the latter algorithm. A distance vector routing algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination. The distance vector routing algorithm is sometimes called by other names, most commonly the distributed BellmanFord routing 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.

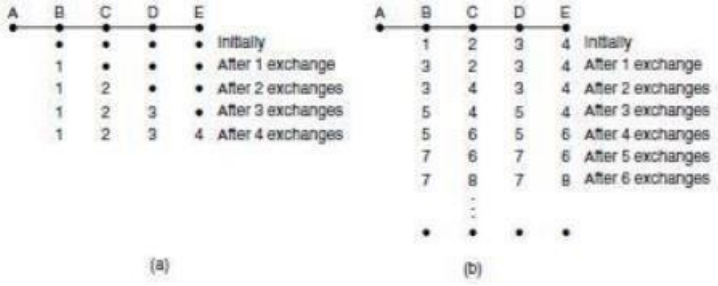
In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network. This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to that destination. The distance might be measured as the number of hops or using another metric, as we discussed for computing shortest paths. The router is assumed to know the ‘‘distance’’ to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is propagation delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can. As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors. Once every T m sec, each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor. Imagine that one of these tables has just come in from neighbor X, with Xi being X’s estimate of how long it takes to get to router
i. If the router knows that the delay to X is m m sec, it also knows that it can reach router i via X in Xi m msec. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding link in its new routing table. Note that the old routing table is not used in the calculation. This updating process is illustrated in Fig. 5-9. Part (a) shows a network. The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K, as 8, 10, 12, and 6 m sec, respectively.



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

Consider how J computes its new route to router G. It knows that it can get to A in 8 m sec, and furthermore A claims to be able to get to G in 18 m sec, so J knows it can count on a delay of 26 m sec to G if it forwards packets bound for G to A. Similarly, it computes the delay to G via I, H, and K as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) m sec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 m sec and that the route to use is via H. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

The Count-to-Infinity Problem: The settling of routes to best paths across the network is called convergence. Distance vector routing is useful as a simple technique by which routers can collectively compute shortest paths, but it has a serious drawback in practice: although it converges to the correct answer, it may do so slowly. In particular, it reacts rapidly to good news, but leisurely to bad news. Consider a router whose best route to destination X is long. If, on the next exchange, neighbor A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X. In one vector exchange, the good news is processed. To see how fast good news propagates, consider the five-node (linear) network of Fig. 5-10, 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.



The count-to-infinity problem.

When A comes up, the other routers learn about it via the vector exchanges. For simplicity, we will assume that there is a gigantic going somewhere that is struck periodically to initiate a vector exchange at all routers simultaneously. At the time of the first exchange, B learns that its lefhand neighbor has zero delay to A. B now makes an entry in its routing table indicating that A is one hop away to the left. All the other routers still think that A is down. At this point, the routing table entries for A are as shown in the second row of Fig. (a). On the next exchange, C learns that B has a path of length 1 to A, so it updates its routing table to indicate a path of length 2, but D and E do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a network whose longest path is of length N hops, within N exchanges everyone will know about newly revived links and routers. Now let us consider the situation of (b), in which all the links and routers are initially up. Routers B, C, D, and E have distances to A of 1, 2, 3, and 4 hops, respectively. Suddenly, either A goes down or the link between A and B is cut (which is effectively the same thing from B's point of view).At the first packet exchange, B does not hear anything from A. Fortunately, C says ''Do not worry; I have a path to

A of length 2.'' Little does B suspect that C's path runs through B itself. For all B knows, C might have ten links all with separate paths to A of length 2. As a result, B thinks it can reach A via C, with a path length of 3. D and E do not update their entries for A on the first exchange.

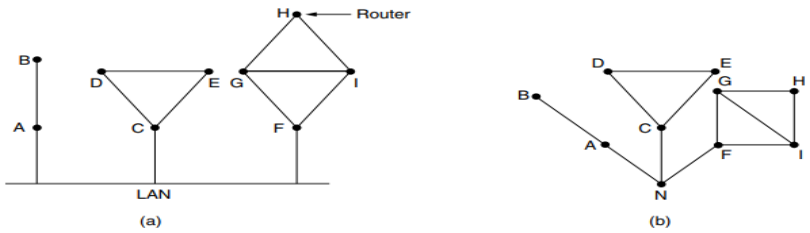
On the second exchange, C notices that each of its neighbors claims to have a path to A of length 3. It picks one of them at random and makes its new distance to A 4, as shown in the third row of Fig. 5-10(b). Subsequent exchanges produce the history shown in the rest of Fig. 5-10(b).From this figure, it should be clear why bad news travels Slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity. For this reason, it is wise to set infinity to the longest path plus 1.Not entirely surprisingly, this problem is known as the count-to-infinity problem. There have been many attempts to solve it, for example, preventing routers from advertising their best paths back to the neighbors from which they heard them with the split horizon with poisoned reverse rule discussed in RFC 1058.However, none of these heuristics work well in practice despite the colorful names. The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path.

Link State Routing: Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem). Consequently, it was replaced by an entirely new algorithm, now called link state routing. Variants of link state routing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today. The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

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

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

Learning about the Neighbors: When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply giving its name. These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F. When two or more routers are connected by a broadcast link (e.g., a switch, ring, or classic Ethernet), the situation is slightly more complicated. Fig. below (a) illustrates a broadcast LAN to which three routers, A, C, and F, are directly connected. Eachof these routers is connected to one or more additional routers, as shown.

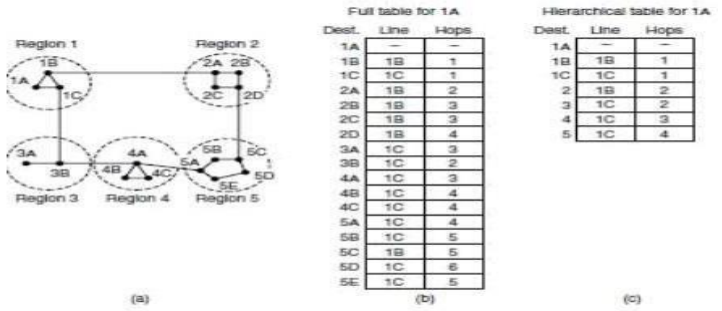


Hierarchical Routing: As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed tosend status reports about them. At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the

telephone network. When hierarchical routing is used, the routers are divided into what we will call regions. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones. 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.

As an example of a multilevel hierarchy, consider how a packet might be routed from Berkeley, California, to Malindi, Kenya. The Berkeley router would know the detailed topology within California but would send all out-of-state traffic to the Los Angeles router. The Los Angeles router would be able to route traffic directly to other domestic routers but would send all foreign traffic to New York. The New York router would be programmed to direct all traffic to the router in the destination country responsible for handling foreign traffic, say, in Nairobi. Finally, the packet would workits way down the tree in Kenya until it got to Malindi. Figure gives a quantitative example of routing in a two-level hierarchy with five regions. The full routing table for router 1A has 17 entries, as shown in Fig. (b). When routing is done hierarchically, as in Fig. 5-14(c), there are entries for all the local routers, as before, but all other regions are condensed into a single router, so all traffic for region 2 goes via the 1B-2A line, but the rest of the remote traffic goes via the 1C-3B line. Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number ofrouters per region grows, the savings in table space increase.

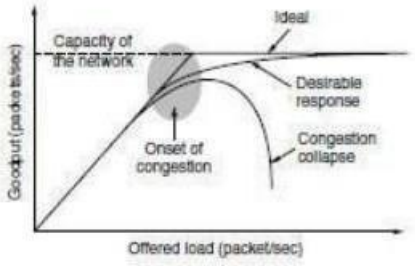
Unfortunately, these gains in space are not free. There is a penalty to be paid: increased path length. For example, the best route from 1A to 5C is via region 2,but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.When a single network becomes very large, an interesting question is ‘‘how many levels should the hierarchy have?’’ For example, consider a network with 720 routers. If there is no hierarchy, each router needs 720 routing table entries. If the network is partitioned into 24 regions of 30 routers each, each router needs 30 local entries plus 23 remote entries for a total of 53 entries. If a three-level hierarchy is chosen, with 8 clusters each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regions within its own cluster, and 7 entries for distant clusters, for a total of 25 entries. Kamoun and Kleinrock (1979) discoveredthat the optimal number of levels for an N router network is ln N, requiring a total of e ln N entries per router. They have also shown that the increase in effective mean path length caused by hierarchical routing is sufficiently small that it is usually acceptable.



Hierarchical routing.

CONGESTION CONTROL ALGORITHMS: Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called congestion. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together. In this chapter we will look at the network aspects of congestion. In Chap. 6, we will complete the topic by covering the transport aspects of congestion. Figure depicts the onset of congestion. When the number of packets hosts send into the network is well within its carrying capacity, the number delivered is proportional to the number sent. If twice as many are sent, twice as many are delivered. However, as the offered load approaches the

carrying capacity, bursts of traffic occasionally fill up the buffers inside routers and some packets are lost. These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is nowcongested. With too much traffic, performance drops sharply.



Unless the network is well designed, it may experience a congestion collapse, in which performance plummets as the offered load increases beyond the capacity. This can happen because packets can be sufficiently delayed inside the network that they are no longer useful when they leave the network. For example, in the early Internet, the time a packet spent waiting for a backlog of packets ahead of it to be sent over a slow 56-kbps link could reach the maximum time it was allowed to remain in the network. It then had to be thrown away.

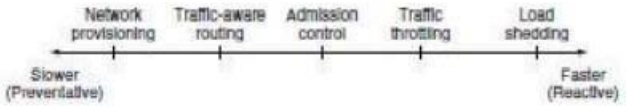
A different failure mode occurs when senders retransmit packets that are greatly delayed, thinking that they have been lost. In this case, copies of the same packet will be delivered by the network, again wasting its capacity. To capture these factors, the y-axis of Fig. is given as good put, which is the rate at which useful packets are delivered by the network. We would like to design networks that avoid congestion where possible and do not suffer from congestion collapse if they do become congested. Unfortunately, congestion cannot wholly be avoided. 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. Adding more memory may help up to a point, but Nagle (1987) realized that if routers have an infinite amount of memory, congestion gets worse, not better. This is because by thetime packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. This makes matters worse, not better—it leads to congestion collapse. Low-bandwidth links or routers that process packets more slowly than the line rate can also become congested.

In this case, the situation can be improved by directing some of the traffic away from the bottleneck to other parts of the network. Eventually, however, all regions of the network will be congested. In this situation, there is no alternative but to shed load or build a faster network. It is worth pointing out the difference between congestion control and flow control, as the relationship is a very subtle one. Congestion control has to do with making sure the network is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts and routers. Flow control, in contrast, relates to the traffic between a particular sender and a particular receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it. To see the difference between these two concepts, consider a network made up of 100- Gbps fiber optic links on which a supercomputer is trying to force feed a large file toa personal computer that is capable of handling only 1 Gbps. Although there is no congestion (the network itself is not in trouble), flow control is needed to force the supercomputer to stop frequently to give the personal computer chance to breathe. At the other extreme, consider a network with 1- Mbps lines and 1000 large computers, half of which are tryingto transfer files at 100 kbps to the other half. Here, the problem is not that of fast senders overpowering slow receivers, butthat the total offered traffic exceeds what the network can handle.

The reason congestion control and flow control are often confused is that the best way to handle both problems is to get the host to slow down. Thus, a host can get a ‘‘slow down’’ message either because the receiver cannot handle the load or because the network cannot handle it. We will come back to this point in Chap. 6. We will start our study of congestion control by looking at the approaches that can be used at different time scales. Then we will look at approaches to preventing congestion from occurring in the first place, followed by approaches for coping with it once it has set in.

Approaches to Congestion Control The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the

load. As shown in Fig., these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.



Timescales of approaches to congestion control.

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. More often, links and routers that are regularly heavily utilized are upgraded at the earliest opportunity. This is called provisioning and happens on a time scale of months, driven by long-term traffic trends. To make the most of the existing network capacity, routes can be tailored to traffic patterns that change during the day as network user's wake and sleep in different time zones. For example, routes may be changed to shift traffic away from heavily used paths by changing the shortest path weights. Some local radio stations have helicopters flying around their cities to report on road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called traffic-aware routing.

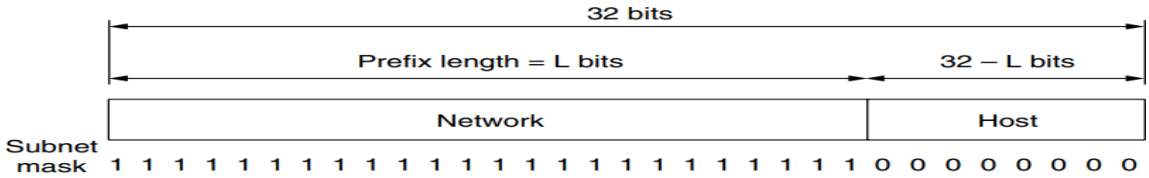
Splitting traffic across multiple paths is also helpful. However, sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called admission control. At a finer granularity, when congestion is imminent the network can deliver feedback to the sources whose traffic flows are responsible for the problem. The network can request these sources to throttle their traffic, or it can slow down the traffic itself. Two difficulties with this approach are how to identify the onset of congestion, and how to inform the source that needs to slowdown. To tackle the first issue, routers can monitor the average load, queuing delay, or packet loss. In all cases, rising numbers indicate growing congestion. To tackle the second issue, routers must participate in a feedback loop with the sources. For a scheme to work correctly, the time scale must be adjusted carefully. If every time two packets arrive in a row, a router yells STOP and every time a router is idle for 20 sec, it yells GO, the system will oscillate wildly and never converge. On the other hand, if it waits 30 minutes to make sure before saying anything, the congestion-control mechanism will react too sluggishly to be of any use. Delivering timely feedback is a nontrivial matter. An added concern is having routers send more messages when the network is already congested. Finally, when all else fails, the network is forced to discard packets that it cannot deliver. The general name for this is load shedding. A good policy for choosing which packets to discard can help to prevent congestion collapse.

IP addresses:

A defining feature of IPv4 is its 32-bit addresses. Every host and router on the Internet has an IP address that can be used in the Source address and Destination address fields of IP packets. It is important to note that an IP address does not actually refer to a host. It really refers to a network interface, so if a host is on two networks, it must have two IP addresses. However, in practice, most hosts are on one network and thus have one IP address. In contrast, routers have multiple interfaces and thus multiple IP addresses. **Prefixes**: IP addresses are hierarchical, unlike Ethernet addresses. Each32-bit address is comprised of a variable-length network portion in the top bits and a host portion in the bottom bits. The network portion has the same value for all hosts on a single network, such as an Ethernet LAN. This means that a network corresponds to a contiguous block of IP address space. This block is called a **prefix**.

IP addresses are written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address 80D00297 is written as 128.208.2.151. Prefixes are written by giving the lowest IP address in the block and the size of the block. The size is determined by the number of bits in the network portion; the remaining bits in the host portion can vary. This means that the size must be a power of two. By convention, itis written after the prefix IP address as a slash followed by the length in bits of the network portion. In our example, if the

prefix contains 28 addresses and so leaves 24 bits for the network portion, it is written as 128.208.0.0/24. Since the prefix length cannot be inferred from the IP address alone, routing protocols must carry the prefixes to routers. Sometimes prefixes are simply described by their length, as in a ‘‘/16’’ which is pronounced ‘‘slash 16.’’ The length of the prefix corresponds to a binary mask of 1s in the network portion. When written out this way, it is called a **subnet mask**. It can beANDed with the IP address to extract only the network portion. For our example, the subnet mask is 255.255.255.0. Fig. below shows a prefix and a subnet mask.



Hierarchical addresses have significant advantages and disadvantages. The key advantage of prefixes is that routers can forward packets based on only the network portion of the address, as long as each of the networks has a uniqueaddress block. The host portion does not matter to the routers because all hosts on the same network will be sent in the same direction. It is only when the packets reach the network for which they are destined that they are forwarded to the correct host. This makes the routing tables much smaller than they would otherwise be. Consider that the number of hosts on the Internet is approaching one billion. That would be a very large table for every router to keep. However, by using a hierarchy, routers need to keep routes for only around 300,000 prefixes.

While using a hierarchy lets Internet routing scale, it has two disadvantages. First, the IP address of a host depends on where it is located in the network. An Ethernet address can be used anywhere in the world, but every IP address belongs to a specific network, and routers will only be able to deliver packets destined to that address to the network. Designs such as mobile IP are needed to support hosts that move between networks but want to keep the same IPaddresses.

The second disadvantage is that the hierarchy is wasteful of addresses unless it is carefully managed. If addresses are assigned to networks in (too) large blocks, there will be (many) addresses that are allocated but not in use. This allocation would not matter much if there were plenty of addresses to go around. However, it was realized more than two decades ago that the tremendous growth of the Internet was rapidly depleting the free address space. IPv6 is the solution to this shortage, but until it is widely deployed there will be great pressure to allocate IP addresses so that they are used very efficiently.

CIDR—Classless InterDomain Routing:

Even if blocks of IP addresses are allocated so that the addresses are used efficiently, there is still a problem that remains: routing table explosion.

Routers in organizations at the edge of a network, such as a university, need to have an entry for each of their subnets, telling the router which line to use to get to that network. For routes to destinations outside of the organization, they can use the simple default rule of sending the packets on the line toward the ISP that connects the organization to the rest of the Internet. The other destination addresses must all be out there somewhere.

Routers in ISPs and backbones in the middle of the Internet have no such luxury. They must know which way to go to get to every network and no simple default will work. These core routers are said to be in the default-free zone of the Internet. No one really knows how many networks are connected to the Internet any more, but it is a large number, probably at least a million. This can make for a very large table. It may not sound large by computer standards, but realize that routers must perform a lookup in this table to forward every packet, and routers at large ISPs may forward up to millions of packets per second. Specialized hardware and fast memory are needed to process packets at these rates, not a generalpurpose computer.

In addition, routing algorithms require each router to exchange information about the addresses it can reach with other routers. The larger the tables, the more information needs to be communicated and processed. The processing grows

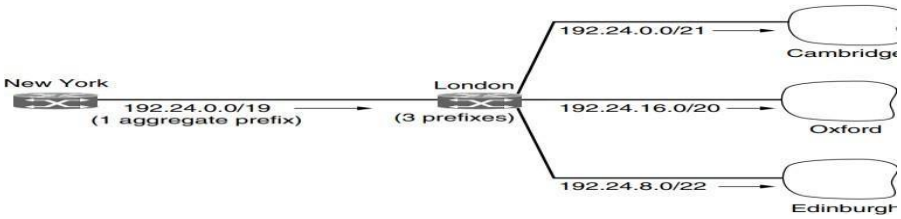
at least linearly with the table size. Greater communication increases the likelihood that some parts will get lost, at least temporarily, possibly leading to routing instabilities.

The routing table problem could have been solved by going to a deeper hierarchy, like the telephone network. For example, having each IP address contain a country, state/province, city, network, and host field might work. Then, each router would only need to know how to get to each country, the states or provinces in its own country, the cities in its state or province, and the networks in its city. Unfortunately, this solution would require considerably more than 32 bits for IP addresses and would use addresses inefficiently (and Liechtenstein would have as many bits in its addresses as the United States). Fortunately, there is something we can do to reduce routing table sizes. We can apply the same insight as subnetting: routers at different locations can know about a given IP address as belonging to prefixes of different sizes. However, instead of splitting an address block into subnets, here we combine multiple small prefixes into a single larger prefix.

To make CIDR easier to understand, let us consider an example in which a block of 8192 IP addresses is available starting at 194.24.0.0. Suppose that Cambridge University needs 2048 addresses and is assigned the addresses 194.24.0.0 through 194.24.7.255, along with mask 255.255.248.0. This is a /21 prefix. Next, Oxford University asks for 4096 addresses. Since a block of 4096 addresses must lie on a 4096-byte boundary, Oxford cannot be given addresses starting at 194.24.8.0. Instead, it gets 194.24.16.0 through 194.24.31.255, along with subnet mask 255.255.240.0. Finally, the University of Edinburgh asks for 1024 addresses and is assigned addresses 194.24.8.0 through 194.24.11.255 and mask 255.255.252.0. These assignments are summarized in Fig.

University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

All of the routers in the default-free zone are now told about the IP addresses in the three networks. Routers close to the universities may need to send on a different outgoing line for each of the prefixes, so they need an entry for each of the prefixes in their routing tables. An example is the router in London in Fig. below. Now let us look at these three universities from the point of view of a distant router in New York. All of the IP addresses in the three prefixes should be sent from New York (or the U.S. in general) to London. The routing process in London notices this and combines the three prefixes into a single aggregate entry for the prefix 194.24.0.0/19 that it passes to the New York router. This prefix contains 8K addresses and covers the three universities and the otherwise unallocated 1024 addresses. By using aggregation, three prefixes have been reduced to one, reducing the prefixes that the New York router must be told about and the routing table entries in the New York router.



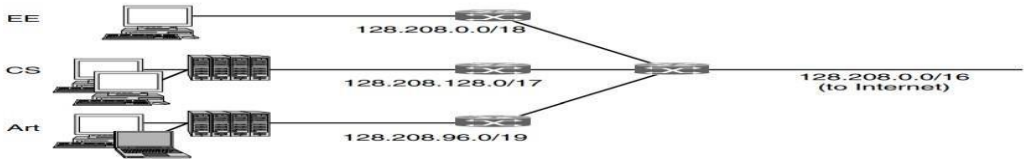
Sub netting: Network numbers are managed by a nonprofit corporation called ICANN (Internet Corporation for Assigned Names and Numbers), to avoid conflicts. In turn, ICANN has delegated parts of the address space to various regional authorities, which dole out IP addresses to ISPs and other companies. This is the process by which a company is allocated a block of IP addresses. However, this process is only the start of the story, as IP address assignment is ongoing as companies grow. We have said that routing by prefix requires all the hosts in a network to have the same network number.This property can cause problems as networks grow. For example, consider a university that started out with our example /16 prefix for use by the Computer Science Dept. for the computers on its Ethernet. A year later, the Electrical Engineering Dept. wants to get on the Internet. The Art Dept. soon follows suit. What IP addresses should these

departments use? Getting further blocks requires going outside the university and may be expensive or inconvenient. Moreover, the /16 already allocated has enough addresses for over 60,000 hosts. It might be intended to allow for significant growth, but until that happens, it is wasteful to allocate further blocks of IP addresses to the same university. A different organization is required. The solution is to allow the block of addresses to be split into several parts for internal use as multiple networks, while still acting like a single network to the outside world. This is called subnetting and the networks (such as Ethernet LANs) that result from dividing up a larger network are called subnets. As we mentioned in Chap. 1, you should be aware that this new usage of the term conflicts with older usage of ‘‘subnet’’ to mean the set of all routers and communication lines in a network. Fig. 5-49 shows how subnets can help with our example. The single /16 has been split into pieces. This split does not need to be even, but each piece must be aligned so that any bits can be usedin the lower host portion. In this case, half of the block (a /17) is allocated to the Computer Science Dept, a quarter is allocated to the Electrical Engineering Dept. (a /18), and one eighth (a /19) to the Art Dept. The remaining eighth is unallocated. A different way to see how the block was divided is to look at the resulting prefixes when written in binary notation:

When a packet comes into the main router, how does the router know which subnet to give it to? This is where the

Computer Science:	10000000	11010000	1|xxxxxxx	xxxxxxxx
Electrical Eng.:	10000000	11010000	00|xxxxxx	xxxxxxxx
Art:	10000000	11010000	011|xxxxx	xxxxxxxx

Here, the vertical bar (|) shows the boundary between the subnet number and the host portion.



details of our prefixes come in. One way would be for each router to have a table with 65,536 entries telling it which outgoing line to use for each host on campus. But this would undermine the main scaling benefit we get from using a hierarchy. Instead, the routers simply need to know the subnet masks for the networks on campus.

When a packet arrives, the router looks at the destination address of the packet and checks which subnet it belongs to. The router can do this by ANDing the destination address with the mask for each subnet and checking to see ifthe result is the corresponding prefix. For example, consider a packet destined for IP address 128.208.2.151. To see if it is for the Computer Science Dept., we AND with 255.255.128.0 to take the first 17 bits (which is 128.208.0.0) and see if they match the prefix address (which is 128.208.128.0). They do not match. Checking the first 18 bits for the Electrical Engineering Dept., we get 128.208.0.0 when ANDing with the subnet mask. This does match the prefix address, so the packet is forwarded onto the interface which leads to the Electrical Engineering network.

The subnet divisions can be changed later if necessary, by updating all subnet masks at routers inside the university. Outside the network, the subnetting is not visible, so allocating a new subnet does not require contacting ICANN or changing any external databases.

Super Netting:

Supernetting is the opposite of subnetting. In subnetting, a single big network is divided into multiple smaller subnetworks. In Supernetting, multiple networks are combined into a bigger network termed as a Supernetwork or Supernet.

Supernetting is mainly used in Route Summarization, where routes to multiple networks with similar network prefixes are combined into a single routing entry, with the routing entry pointing to a Super network, encompassing all the networks. This in turn significantly reduces the size of routing tables and also the size of routing updates exchanged by routing protocols.

- When multiple networks are combined to form a bigger network, it is termed super-netting
- Super netting is used in route aggregation to reduce the size of routing tables and routing table updates

There are some points which should be kept in mind while supernetting:
All the Networks should be contiguous.
The block size of every network should be equal and must be in form of 2^n.

1. First Network id should be exactly divisible by whole size of supernet.

Example – Suppose 4 small networks of class C:

200.1.0.0,

200.1.1.0,

200.1.2.0,

200.1.3.0

Build a bigger network that has a single Network Id.

Explanation – Before Supernetting routing table will look like as:

Network Id	Subnet Mask	Interface
200.1.0.0	255.255.255.0	A
200.1.1.0	255.255.255.0	B
200.1.2.0	255.255.255.0	C
200.1.3.0	255.255.255.0	D

First, let's check whether three conditions are satisfied or not:

1. **Contiguous:** You can easily see that all networks are contiguous all having size 256 hosts.
   Range of first Network from 200.1.0.0 to 200.1.0.255. If you add 1 in last IP address of first network that is 200.1.0.255 + 0.0.0.1, you will get the next network id which is 200.1.1.0. Similarly, check that all network are contiguous.
2. **Equal size of all network:** As all networks are of class C, so all of them have a size of 256 which is in turn equalto 2^8.
3. **First IP address exactly divisible by total size:** When a binary number is divided by 2^n then last n bits are the remainder. Hence in order to prove that first IP address is exactly divisible by while size of Supernet Network. You can check that if last n v=bits are 0 or not.
   In the given example first IP is 200.1.0.0 and whole size of supernet is $4*2^8 = 2^{10}$. If last 10 bits of first IP address are zero then IP will be divisible.

11001000	00000001	00000000	00000000
200	1	0	0

Last 10 bits of first IP address are zero (highlighted by green color). So 3rd condition is also satisfied.

1. Control and reduce network traffic

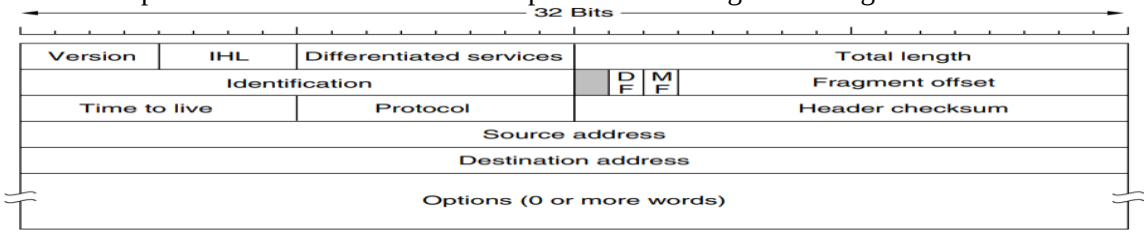
2. Helpful to solve the problem of lacking IP addresses

3. Minimizes the routing table
   - It cannot cover a different area of the network when combined
   - All the networks should be in the same class and all IP should be contiguous

The IP Version 4 Protocol(IPv4):

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 5-46. The bits are transmitted from left to right andtop to bottom, with the high-order bit of the Version field going first. (This is a ‘‘big-endian’’ network byte order. On littleendian machines, such as Intel x86 computers, a software conversion is required on both transmission and reception.) In retrospect, little endian would have been a better choice, but at the time IP was designed, no one knew it would come todominate computing.

The Version field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet



today, and that is where we have started our discussion. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time. In fact, IPv6, the next version of IP, was defined more than a decade ago, yet is only just beginning to be deployed. We will describe it later in this section. Its use will eventually be forced when each of China’s almost 231 people has a desktop PC, a laptop, and an IP phone. As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used.

Since the header length is not constant, a field in the header, IHL, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the Options field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless.

The Differentiated services field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the Type of service field. It was and still is intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission. The Type of service field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability. However, no one really knew what to do with these bits at routers, so they were left unused for many years. When differentiated services were designed, IETF threw in the towel and reused this field. Now, the top 6 bits are used to mark the packet with its service class; we described the expedited and assured services earlier in this chapter. The bottom 2 bits are used to carry explicit congestion notification information, such as whether the packet has experienced congestion; we described explicit congestion notification as part of congestion control earlier in this chapter.

The Total length includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future networks, larger datagrams may be needed.

The Identification field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same Identification value.

Next comes an unused bit, which is surprising, as available real estate in the IP header is extremely scarce. As an April fool's joke, Bellovin (2003) proposed using this bit to detect malicious traffic. This would greatly simplify security, as packets with the ''evil'' bit set would be known to have been sent by attackers and could just be discarded. Unfortunately, network security is not this simple.

Then come two 1-bit fields related to fragmentation. DF stands for Don't Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again. Now it is used as part of the process to discover the path MTU, which is the largest packet that can travel along a path without being fragmented. By marking the datagram with the DF bit, the sender knows it will either arrive in one piece, oran error message will be returned to the sender.

MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The Fragment offset tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the Total length field.

The TtL (Time to live) field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents packets from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete packet, it needs to know what to do with it. The Protocol field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet. Protocols and other assigned numbers were formerly listed in RFC 1700, but nowadays they are contained in an online database located at www.iana.org.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the Header checksum. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the Header checksum is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network. Note that it must be recomputed at each hop because at least one field always changes (the Time to live field), but tricks can be used to speed up the computation.

The Source address and Destination address indicate the IP address of the source and destination network interfaces. We will discuss Internet addresses in the next section.

The Options field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length field, and then one or more data bytes. The Options field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig. 5-47 were defined.

Packet Fragmentation: Each network or link imposes some maximum size on its packets. These limits have various causes, among them

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish. Maximum payloads for some common technologies are 1500 bytes for Ethernet and 2272 bytes for 802.11. IP is more generous, allows for packets as big as 65,515 bytes. Hosts usually prefer to transmit large packets because this reduces

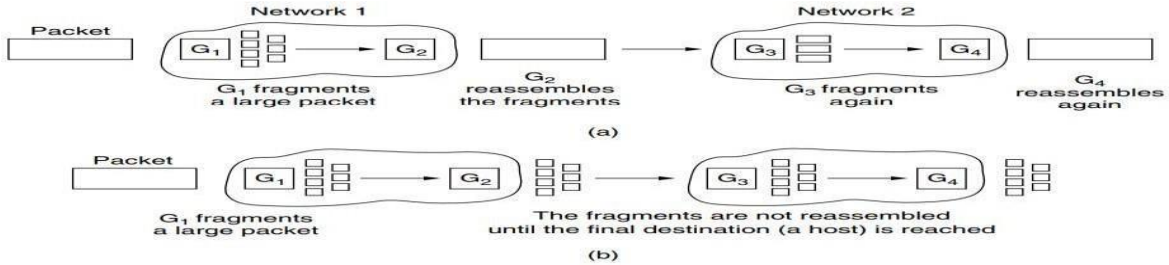
packet overheads such as bandwidth wasted on header bytes. An obvious internetworking problem appears when a large packet wants to travel through a network whose maximum packet size is too small. This nuisance has been a persistent issue, and solutions to it have evolved along with much experience gained on the Internet. One solution is to make sure the problem does not occur in the first place. However, this is easier said than done. A source does not usually know the path a packet will take through the network to a destination, so it certainly does not know how small packets must be to get there. This packet size is called the Path MTU (Path Maximum Transmission Unit).

Even if the source did know the path MTU, packets are routed independently in a connectionless network such as the Internet. This routing means that paths may suddenly change, which can unexpectedly change the path MTU. The alternative solution to the problem is to allow routers to break up packets into fragments, sending each fragment as a separate network layer packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process. (Physicists have even given this effect a name: the second law of thermodynamics.)

Packet-switching networks, too, have trouble putting the fragments back together again. Two opposing strategies exist for recombining the fragments back into the original packet. The first strategy is to make fragmentation caused by a ‘‘smallpacket’’ network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig. 5- 42(a). In this approach, when an oversized packet arrives at G1, the router breaks it up into fragments. Each fragment is addressed to the same exit router, G2, where the pieces are recombined. In this way, passage through the small-packet network is made transparent. Subsequent networks are not even aware that fragmentation has occurred. Transparent fragmentation is straightforward but has some problems. For onething, the exit router must know when it has received all the pieces, so either a count field or an ‘‘end of packet’’ bit must be provided. Also, because all packets must exit via the same router so that they can be reassembled, the routes are constrained. By not allowing some fragments to follow one route to the ultimate destination and other fragments a disjoint route, some performance may be lost.

More significant is the amount of work that the router may have to do. It may need to buffer the fragments as they arrive, and decide when to throw them away if not all of the fragments arrive. Some of this work may be wasteful, too, as the packet may pass through a series of small packet networks and need to be repeatedly fragmented and reassembled. The other fragmentation strategy is to refrain from recombining fragments at any intermediate routers. Once a packet has been fragmented, each fragment is

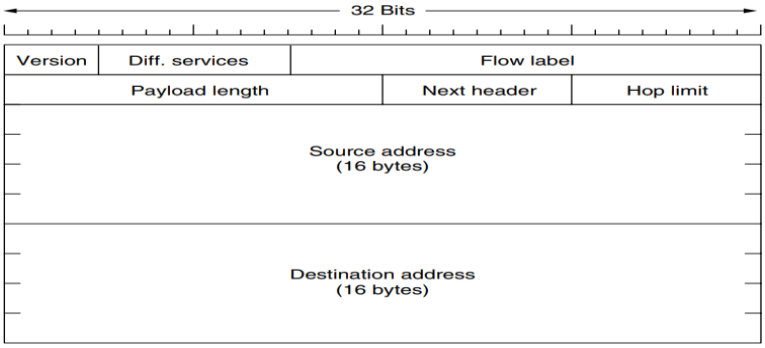
treated as though it were an original packet. The routers pass the fragments, as shown in Fig.shown below, and reassembly is performed only at the destination host.



IP Version 6 IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses.Even 456 THE NETWORK LAYER CHAP. 5 with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012. This looming disaster was recognized almost two decades ago, and it sparked a great deal of discussion and controversy within the Internet community about what to do about it. In this section, we will describe both the problem and several proposed solutions. The only long-term solution is to move to larger addresses. IPv6 (IP version 6) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case. The result is that IPv6 is deployed and

used on only a tiny fraction of the Internet (estimates are 1%) despite having been an Internet Standard since 1998. The next several years will be an interesting time, as the few remaining IPv4 addresses are allocated. Will people start to auction off their IPv4 addresses on eBay? Will a black market in them spring up? Who knows.

The Main IPv6 Header: The IPv6 header is shown in Fig. 5-56. The Version field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have. As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for demultiplexing, so some routers may skip the check. For example, the Ethernet Type field has different values to indicate an IPv4 or an IPv6 payload. The discussions between the ‘‘Do it right’’ and ‘‘Make it fast’’ camps will no doubt be lengthy and vigorous.

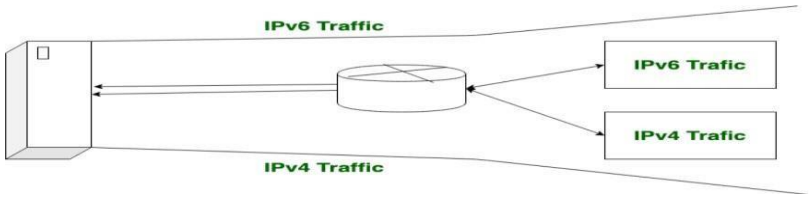


Transition from IPv4 to IPv6 address:

The IPv4 address is exhausted and IPv6 had come to overcome the limit. Various organization is currently working with IPv4 technology and in one day we can't switch directly from IPv4 to IPv6. Instead of only using IPv6, we use combination of both and transition means not replacing IPv4 but co-existing of both.

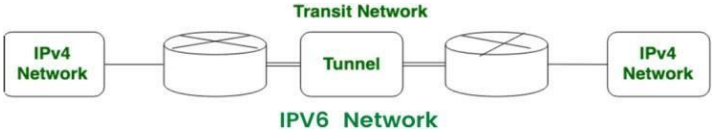
When we want to send a request from an IPv4 address to an IPv6 address, but it isn't possible because IPv4 and IPv6 transition is not compatible. For a solution to this problem, we use some technologies. These technologies are *Dual Stack Routers, Tunneling, and NAT Protocol Translation*. These are explained as following below.

1. Dual-Stack Routers: In dual-stack router, A router's interface is attached with IPv4 and IPv6 addresses configured are used in order to transition from IPv4 to IPv6.



2. Tunneling:

Tunneling is used as a medium to communicate the transit network with the different IP versions.



In this above diagram, the different IP versions such as IPv4 and IPv6 are present. The IPv4 networks can communicate with the transit or intermediate network on IPv6 with the help of the Tunnel. It's also possible that the IPv6 network can also communicate with IPv4 networks with the help of a Tunnel.

3. NAT Protocol Translation: With the help of the NAT Protocol Translation technique, the IPv4 and IPv6 networks can also communicate with each other which do not understand the address of different IP version.

Generally, an IP version doesn't understand the address of different IP version, for the solution of this problem we use NAT-PT device which removes the header of first (sender) IP version address and add the second (receiver) IP version address so that the Receiver IP version address understand that the request is sent by the same IP version, and its vice-versa is also possible.

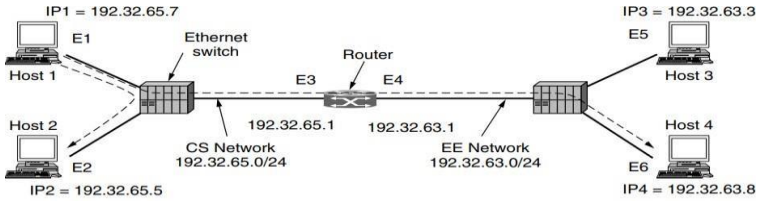
In the above diagram, an IPv4 address communicates with the IPv6 address via a NAT-PT device to communicate easily. In this situation, the IPv6 address understands that the request is sent by the same IP version (IPv6) and it responds.

ARP—The Address Resolution Protocol: Although every machine on the Internet has one or more IP addresses, these addresses are not sufficient for sending packets. Data link layer NICs (Network Interface Cards) such as Ethernet cards do not understand Internet addresses. In the case of Ethernet, every NIC ever manufactured comes equipped with a unique 48-bit Ethernet address. Manufacturers of Ethernet NICs request a block of Ethernet addresses from IEEE to ensure that no two NICs have the same address (to avoid conflicts should the two NICs ever appear on the same LAN). The NICs send and receive frames based on 48-bit Ethernet addresses. They know nothing at all about 32-bit IP addresses.

The question now arises, how do IP addresses get mapped onto data link layer addresses, such as Ethernet? To explain how this works, let us use the example of Fig. 5-61, in which a small university with two /24 networks is illustrated. One network (CS) is a switched Ethernet in the Computer Science Dept. It has the prefix 192.32.65.0/24. The other LAN (EE), also switched Ethernet, is in Electrical Engineering and has the prefix 192.32.63.0/24. The two LANs are connected by an IP router. Each machine on an Ethernet and each interface on the router has a unique Ethernet address, labeled E1 through E6, and a unique IP address on the CS or EE network.

Let us start out by seeing how a user on host 1 sends a packet to a user on host 2 on the CS network. Let us assume the sender knows the name of the intended receiver, possibly something like eagle.cs.uni.edu. The first step is to find the IP address for host 2. This lookup is performed by DNS, which we will study in Chap. 7. For the moment, we will just assume that DNS returns the IP address for host 2 (192.32.65.5).

The upper layer software on host 1 now builds a packet with 192.32.65.5 in the Destination address field and gives it to the IP software to transmit. The IP software can look at the address and see that the destination is on the CS network, (i.e., its own network). However, it still needs some way to find the destination's Ethernet address to send the frame. One solution is to have a configuration file somewhere in the system that maps IP addresses onto Ethernet addresses.



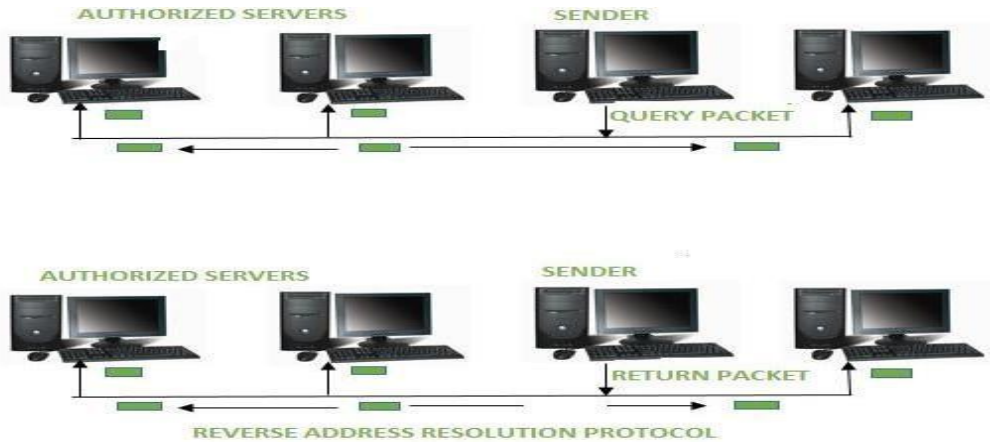
Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6

This solution is certainly possible, for organizations with thousands of machines keeping all these files up to date is an error-prone, time-consuming job. A better solution is for host 1 to output a broadcast packet onto the Ethernet asking who owns IP address 192.32.65.5. The broadcast will arrive at every machine on the CS Ethernet, and each one will check its IP address. Host 2 alone will respond with its Ethernet address (E2). In this way host 1 learns that IP address 192.32.65.5 is on the host with Ethernet address E2. The protocol used for asking this question and getting the reply is called ARP (Address Resolution Protocol). Almost every machine on the Internet runs it. ARP is defined in RFC 826.

What is RARP ?

RARP is abbreviation of **Reverse Address Resolution Protocol** which is a protocol based on computer networking which is employed by a client computer to request its IP address from a gateway server's Address Resolution Protocol

table or cache. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address.
This protocol is used to communicate data between two points in a server. The client doesn't necessarily need prior knowledge the server identities capable of serving its request. Media Access Control (MAC) addresses requires individual configuration on the servers done by an administrator. RARP limits to the serving of IP addresses only.
When a replacement machine is set up, the machine may or might not have an attached disk that may permanently store the IP Address so the RARP client program requests IP Address from the RARP server on the router. The RARP server will return the IP address to the machine under the belief that an entry has been setup within the router table.



History of RARP :
RARP was proposed in 1984 by the university Network group. This protocol provided the IP Address to the workstation. These diskless workstations were also the platform for the primary workstations from Sun Microsystems.
Working of RARP :
The RARP is on the Network Access Layer and is employed to send data between two points in a very network.
Each network participant has two unique addresses:- IP address (a logical address) and MAC address (the physical address).
The IP address gets assigned by software and after that the MAC address is constructed into the hardware.
The RARP server that responds to RARP requests, can even be any normal computer within the network. However, it must hold the data of all the MAC addresses with their assigned IP addresses. If a RARP request is received by the network, only these RARP servers can reply to it. The info packet needs to be sent on very cheap layers of the network. This implies that the packet is transferred to all the participants at the identical time.

The client broadcasts a RARP request with an Ethernet broadcast address and with its own physical address. The server responds by informing the client its IP address.

How is RARP different from ARP ?

RARP	ARP
RARP stands for Reverse Address Resolution Protocol	ARP stands for Address Resolution Protocol
In RARP, we find our own IP address	In ARP, we find the IP address of a remote machine
The MAC address is known and the IP address is requested	The IP address is known, and the MAC address is being requested

RARP	ARP
It uses the value 3 for requests and 4 for responses	It uses the value 1 for requests and 2 for responses

Uses of RARP :

RARP is used to convert the Ethernet address to an IP address. It is available for the LAN technologies like FDDI, token ring LANs, etc.

Disadvantages of RARP :

The Reverse Address Resolution Protocol had few disadvantages which eventually led to its replacement by BOOTP and DHCP. Some of the disadvantages are listed below:

- The RARP server must be located within the same physical network.
- The computer sends the RARP request on very cheap layer of the network. Thus, it's unattainable for a router to forward the packet because the computer sends the RARP request on very cheap layer of the network.
- The RARP cannot handle the subnetting process because no subnet masks are sent. If the network is split into multiple subnets, a RARP server must be available with each of them.
- It isn't possible to configure the PC in a very modern network.
- It doesn't fully utilize the potential of a network like Ethernet.

RARP has now become an obsolete protocol since it operates at low level. Due to this, it requires direct address to the network which makes it difficult to build a server.

Introduction

The transport layer is the core of the OSI model. Protocols at this layer oversee the delivery of data from an application program on one device to an application program on another device. They act as a liaison between the upper-layer protocols (session, presentation, and application) and the services provided by the lower layers.

Duties of the transport layer:

The services provided are similar to those of the data link layer. The data link layer, however, is designed to provide its services within a single network, while the transport layer provides these services across an internetwork made of many networks. While the transport layer controls all three of the lower layers.

The services provided by transport layer protocols can be divided into five broad categories: end- to-end deliver, addressing, reliable delivery, flow control, and multiplexing.

- To provide reliable, cost effective data transfer from source to destination
- This layer deals with end to end transfer of data
- Here transport entity deals with other host's transport entity.
- Transport layers deals with processes running on the host.

Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective service to its users, normally processes in the application layer. To achieve this goal, the transport layer makes use of the services provided by the network layer. The hardware and/or software 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 separate user process, in a library package bound into network applications, or conceivably on the network interface card. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 4-1.

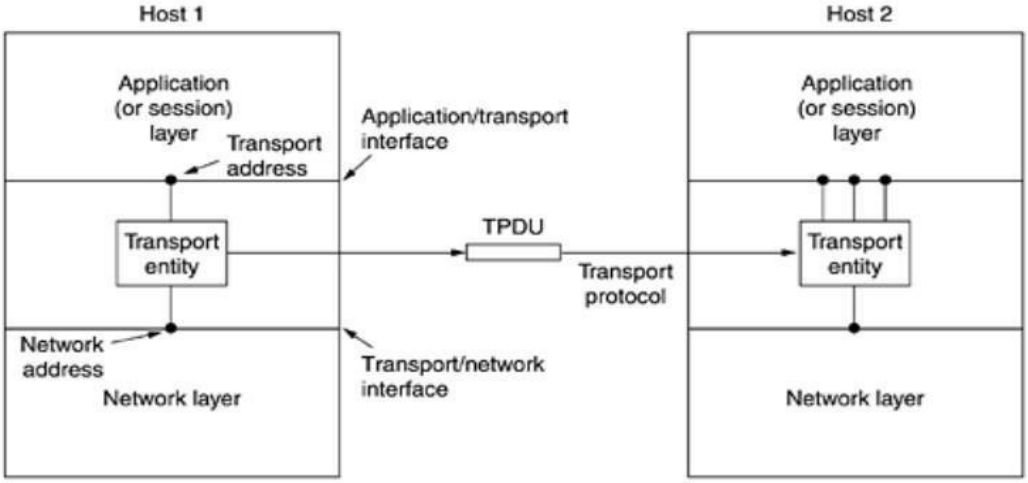


Figure 4-1. The network, transport, and application layers.

The transport code runs entirely on the users' machines, but the network layer mostly runs on the routers, which are operated by the carrier (at least for a wide area network).

In essence, the existence of the transport layer makes it possible for the transport service to be more reliable than the underlying network service. Lost packets and mangled data can be detected and compensated for by the transport layer. Furthermore, the transport service primitives can be implemented as calls to library procedures in order to make them independent of the network service primitives. The

network service calls may vary considerably from network to network (e.g., connectionless LAN service may be quite different from connection-oriented WAN service). By hiding the network service behind a set of transport service primitives, changing the network service merely requires replacing one set of library procedures by another one that does the same thing with a different underlying service.

For this reason, many people have traditionally made a distinction between layers 1 through 4 on the one hand and layer(s) above 4 on the other. The bottom four layers can be seen as the transport service provider, whereas the upper layer(s) are the transport service user. This distinction of provider versus user has a considerable impact on the design of the layers and puts the transport layer in a key position, since it forms the major boundary between the provider and user of the reliable data transmission service.

Transport Service Primitives

To get an idea of what a transport service might be like, consider the five primitives listed in Fig. 4-2. This transport interface is truly bare bones, but it gives the essential flavor of what a connection-oriented transport interface has to do. It allows application programs to establish, use, and then release connections, which is sufficient for many applications.

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

Figure 4-2. The primitives for a simple transport service.

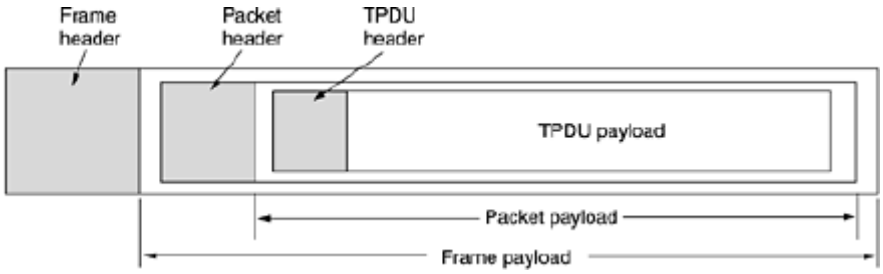


Figure 4-3. Nesting of TPDUs, packets, and frames.

Elements of Transport Protocols

- **Addressing**
- **Connection Establishment**
- **Connection Release**
- **Flow Control and Buffering**
- **Multiplexing**
- **Crash Recovery**

The transport service is implemented by a transport protocol used between the two transport entities. In some ways, transport protocols resemble the data link protocols .Both have to deal with error control, sequencing, and flow control, among other issues.

However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 5-4. At the

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

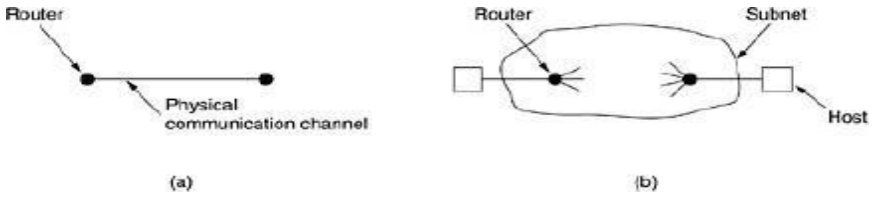


Figure 4-4. (a) Environment of the data link layer. (b) Environment of the transport layer.

Addressing

- **Application Process is connected to the TSAP**
- **Entity connects to the NSAP.**
- **There are multiple processes running within the host.**

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. (Connectionless transport has the same problem: To whom should each message be sent?) The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these end points are called **ports**. In ATM networks, they are called **AAL-SAPs**. We will use the generic term **TSAP**, (**Transport Service Access Point**). The analogous end points in the network layer (i.e., network layer addresses) are then called **NSAPs**. IP addresses are examples of NSAPs.

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.

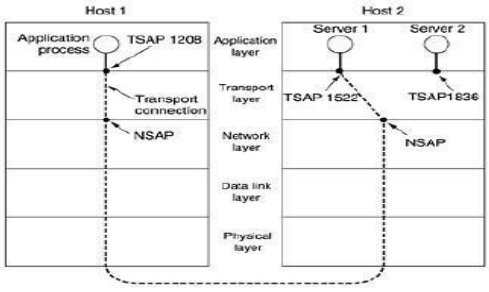


Figure 4-5. TSAPs, NSAPs, and transport connections.

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST TPDU to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, store, and duplicate packets.

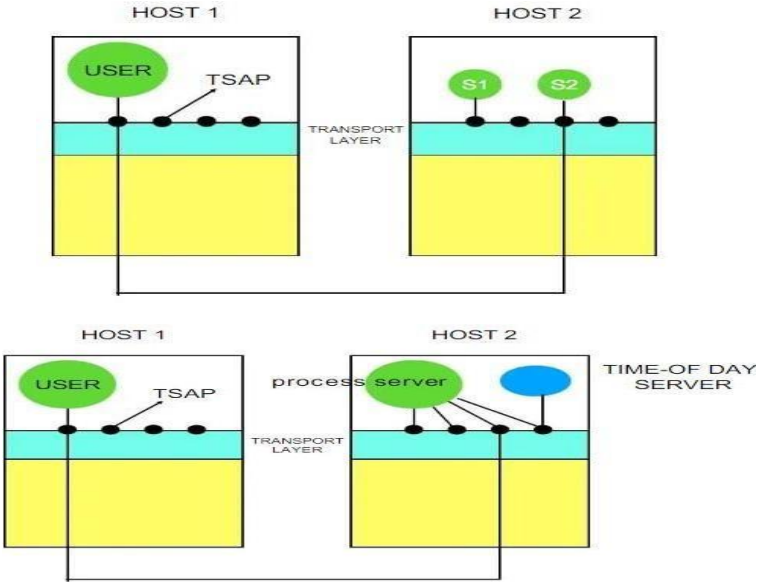


Figure 4-6. How a user process in host 1 establishes a connection with a time-of-day server in host 2.

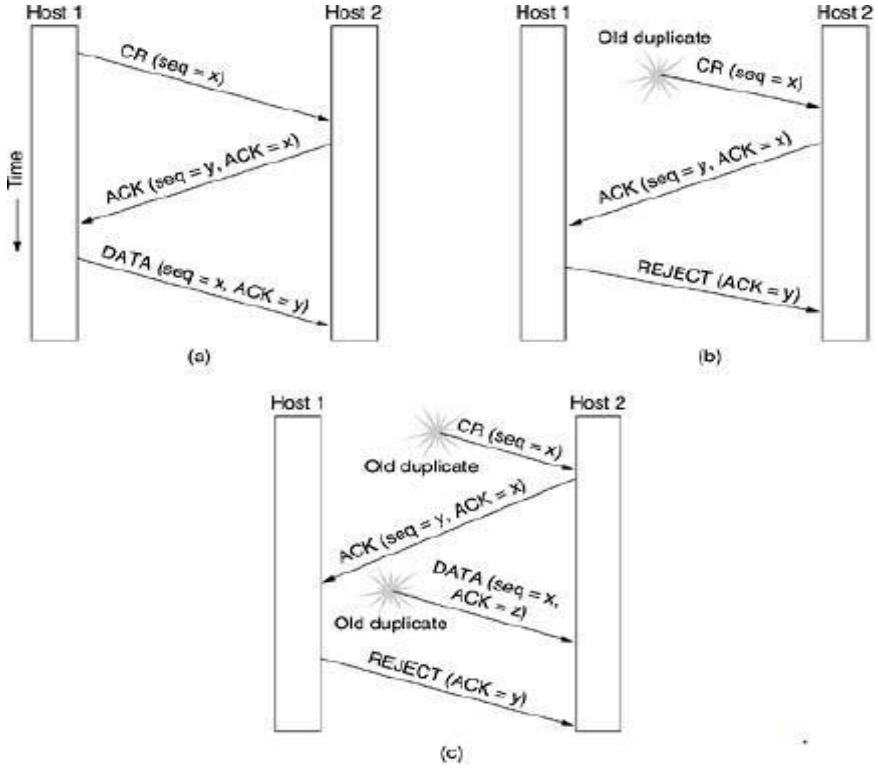


Figure 4-7. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

Connection Release

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.

Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. 4-8. After the connection is established, host 1 sends a TPDU that arrives properly at host 2. Then host 1 sends another TPDU. Unfortunately, host 2 issues a DISCONNECT before the second TPDU arrives. The result is that the connection is released and data are lost.

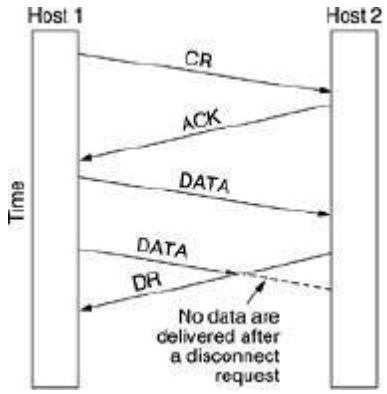


Figure 4-8. Abrupt disconnection with loss of data.

Error Control

The Transmission Control Protocol is designed to provide reliable data transfer between a pair of devices on an IP internetwork. Much of the effort required to ensure reliable delivery of data segments is of necessity focused on the problem of ensuring that data is not lost in transit. But there's another important critical impediment to the safe transmission of data: the risk of errors being introduced into a TCP segment during its travel across the internetwork.

Checksum

To provide basic protection against errors in transmission, TCP includes a 16-bit Checksum field in its header. The idea behind a checksum is very straight-forward: take a string of data bytes and add them all together. Then send this sum with the data stream and have the receiver check the sum. In TCP, a special algorithm is used to calculate this checksum by the device sending the segment; the same algorithm is then employed by the recipient to check the data it received and ensure that there were no errors.

The checksum calculation used by TCP is a bit different than a regular checksum algorithm. A conventional checksum is performed over all the bytes that the checksum is intended to protect, and can detect most bit errors in any of those fields.

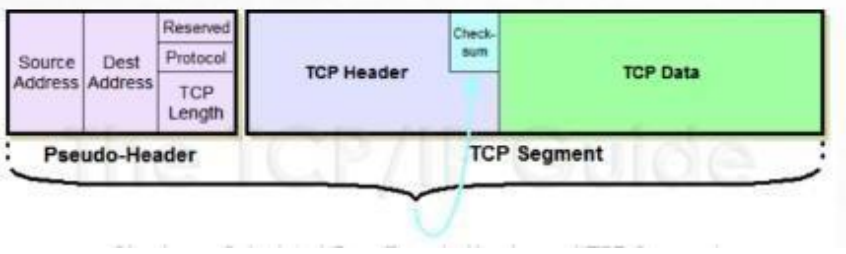


Figure 4-9. Checksum calculated over Pseudo Header and TCP Segment

Acknowledgement

- Cumulative Acknowledgement
- Selective Acknowledgement:

Retransmission

Retransmission, essentially identical with Automatic repeat request (ARQ), is the resending of packets which have been either damaged or lost. Retransmission is a very simple concept. Whenever one party sends something to the other party, it retains a copy of the data it sent until the recipient has acknowledged that it received it. In a variety of circumstances the sender automatically retransmits the data using the retained copy.

Flow Control and Buffering

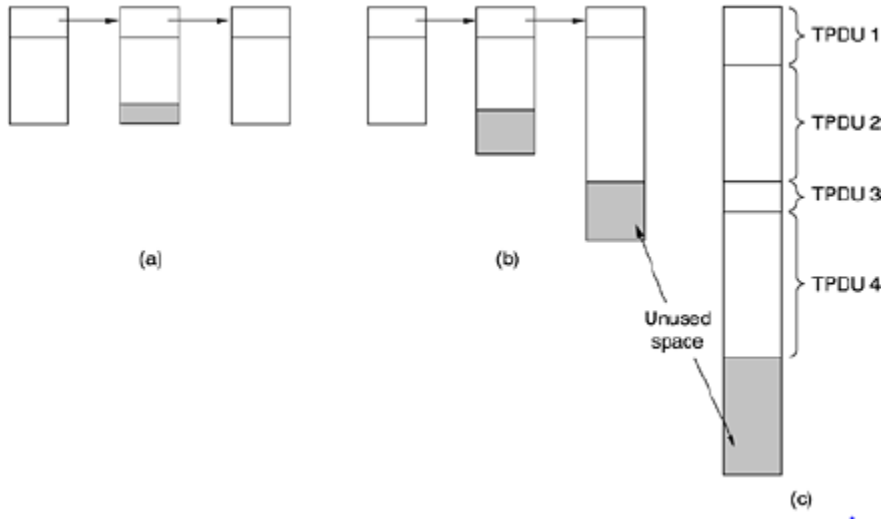


Figure 4-10. (a) Chained fixed-size buffers. (b) Chained variable-sized buffers. (c) One large circular buffer per connection.

Multiplexing

Multiplexing several conversations onto connections, virtual circuits, and physical links plays a role in several layers of the network architecture. In the transport layer the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it. When a TPDU comes in, some way is needed to tell which process to give it to. This situation, called **upward multiplexing**, is shown in Fig. 4-11(a). In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.

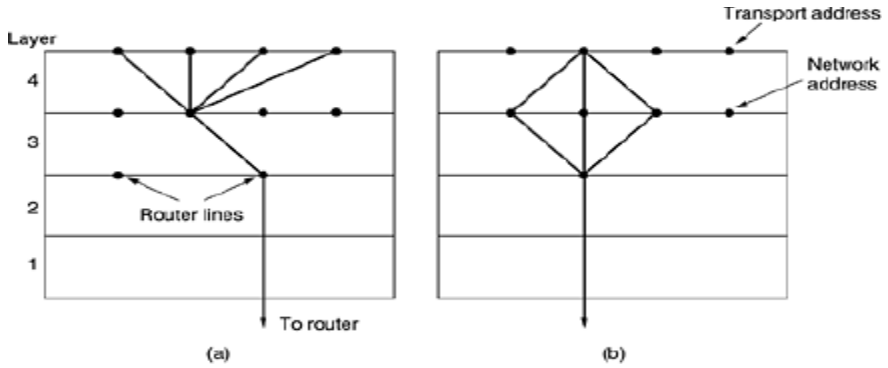


Figure 4-11. (a) Upward multiplexing. (b) Downward multiplexing.

Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a subnet uses virtual circuits internally and imposes a maximum data rate on each one. If a user needs more bandwidth than one virtual circuit can provide, a way out is toopen multiple network connections and distribute the traffic among them on a round-robin basis, as indicated in Fig. 4-11(b). This modus operandi is called **downward multiplexing**. With *k* network connections open, the effective bandwidth is increased by a factor of *k*. A common example of downward multiplexing occurs with home users who have an ISDN line. This line provides for two separate connections of 64 kbps each. Using both of them to call an Internet provider and dividing the traffic over both lines makes it possible to achieve an effective bandwidth of 128 kbps.

Crash Recovery

- Always Retransmit
- First Acknowledgment then write
- Retransmit in S0(Sent 2 messages, Ack of both received).No outstanding Packet is present retransmit inS1(Sent 2 messages, Ack of only 1received). Here Outstanding Packet is present

In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections. Each client can be in one of two states: one TPDU outstanding, *S1*, or no TPDUs outstanding, *S0*. Based on only this state information, the client must decide whether to retransmit the most recent TPDU.

The Internet Transport Protocols: UDP

The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. In the following sections we will study both of them. The connectionless protocol is UDP.The connection-oriented protocol is TCP. Because UDP is basically just IP with a short header added, we will start with it. We will also look at two applications of UDP.

Introduction to UDP

The Internet protocol suite supports a connectionless transport protocol, **UDP** (**User Datagram Protocol**). UDP provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection. UDP is described in RFC 768.

UDP transmits segments consisting of an 8-byte header followed by the payload. The header is shown in Fig. 4-12. 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.

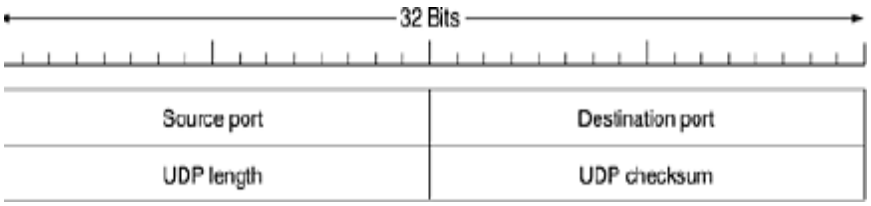


Figure 4-12. The UDP header.

- **Source** port number, which is the number of the sender;
- **Destination** port number, the port to which the datagram is addressed;
- **Length**, the length in bytes of the UDP header and any encapsulated data; and
- **Checksum**, which is used in error checking -- its use is required in IPv6 and optional in IPv4.

The Internet Transport Protocols: TCP

UDP is a simple protocol and it has some niche uses, such as client-server interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed. UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet. Let us now study it in detail.

Introduction to TCP

TCP (Transmission Control Protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.

TCP was formally defined in RFC 793. As time went on, various errors and inconsistencies were detected, and the requirements were changed in some areas. These clarifications and some bug fixes are detailed in RFC 1122. Extensions are given in RFC 1323.

The TCP Service Model

TCP service is obtained by both the sender and receiver creating end points, called sockets. Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a port. A port is the TCP name for a TSAP. For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine.

A socket may be used for multiple connections at the same time. In other words, two or more connections may terminate at the same socket. Connections are identified by the socket identifiers at ends that is, (*socket1*, *socket2*). No virtual circuit numbers or other identifiers are used.

All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points. TCP does not support multicasting or broadcasting**.**

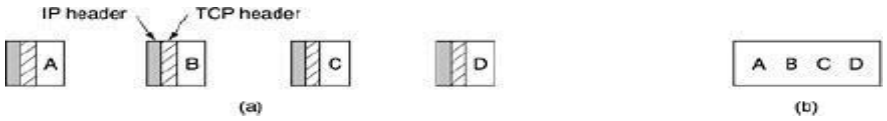


Figure 4-13. (a) Four 512-byte segments sent as separate IP datagrams. (b) The 2048 bytes of data delivered to the application in a single READ call.

The TCP Segment Header

Figure 6-29. The TCP header.

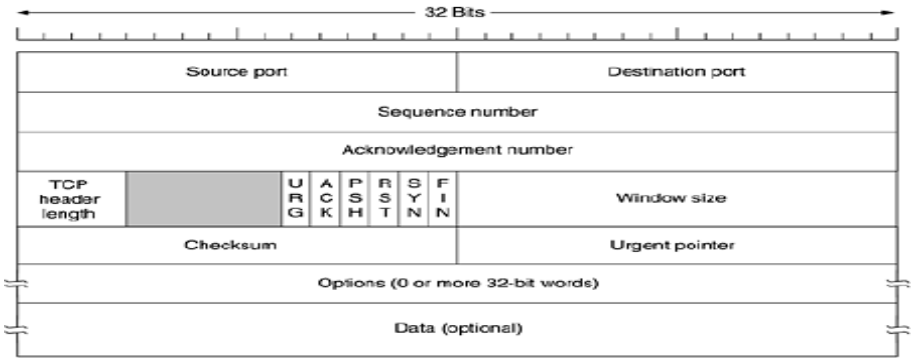


Figure 4-14. The TCP header.

1. Source Port: It is a 16-bit source port number used by the receiver to reply.
2 . Destination Port: It is a 16-bit destination port number.
3. **Sequence Number:** The sequence number of the first data byte in this segment. During the SYN Control bit is set, and the sequence number is n, and the first data byte is n + 1.
4. **Acknowledgement Number:** If the ACK control bit is set, this field contains the next number that the receiver expects to receive.
5. **Data Offset:** The several 32-bit words in the TCP header shows from where the user data begins.
Reserved (6 bit): It is reserved for future use.
URG: It indicates an urgent pointer field that data type is urgent or not.
ACK: It indicates that the acknowledgement field in a segment is significant, as discussed early.
PUSH: The PUSH flag is set or reset according to a data type that is sent immediately or not.
RST: It Resets the connection.
SYN: It synchronizes the sequence number.

FIN: This indicates no more data from the sender.
Window: It is used in Acknowledgement segment. It specifies the number of data bytes, beginning with the one indicated in the acknowledgement number field that the receiver is ready to accept.
Checksum: It is used for error detection.
Options: The IP datagram options provide additional punctuality. It can use several optional parameters between a TCP sender and receiver. It depends on the options used. The length of the field may vary in size, but it can't be larger than 40 bytes due to the header field's size, which is 4 bit.

TCP Connection Establishment

Connections are established in TCP by means of the three-way handshake. 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.

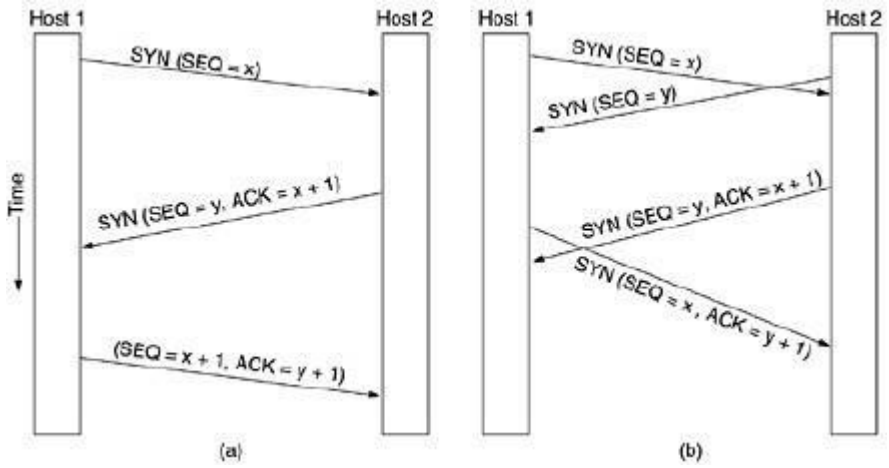


Figure 4-15. (a) TCP connection establishment in the normal case. (b) Call collision.

When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port given in the *Destination port* field. If not, it sends a reply with the *RST* bit on to reject the connection.

If some process is listening to the port, that process is given the incoming TCP segment. It can then either accept or reject the connection. If it accepts, an acknowledgement segment is sent back. The sequence of TCP segments sent in the normal case is shown in Fig. 4-14(a). Note that a *SYN* segment consumes 1 byte of sequence space so that it can be acknowledged unambiguously.

In the event that two hosts simultaneously attempt to establish a connection between the same two sockets, the sequence of events is as illustrated in Fig. 4-14(b). The result of these events is that just one connection is established, not two because connections are identified by their end points. If the first setup results in a connection identified by (*x*, *y*) and the second one does too, only one table entry is made, namely, for (*x*, *y*).

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.

Just as with telephone calls in which both people say goodbye and hang up the phone simultaneously, both ends of a TCP connection may send *FIN* segments at the same time. These are each acknowledged in the usual way, and the connection is shut down. There is, in fact, no essential difference between the two hosts releasing sequentially or simultaneously.

The TCP Sliding Window

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the data link layer (OSI layer 2) as well as in the Transmission Control Protocol (TCP). They are also used to improve efficiency when the channel may include high latency.

Sliding window is also known as windowing. A sliding window is a method for controlling sending data packets between two network devices where dependable and sequential delivery of data packets is needed, such as using the Data Link Layer (OSI model) or Transmission Control Protocol (TCP).

In the sliding window technique, each data packet (for most data link layers) and byte (in TCP) includes a unique consecutive sequence number used by the receiving computer to place data in the correct order. The objective of the sliding window technique is to use the sequence numbers to avoid duplicate data and to request missing data.

Following are the two types of Sliding Window Protocol −

- Go Back-n Protocol
- Selective Repetitive ARQ

TCP Congestion Control

In Fig.4-16, we see this problem illustrated hydraulically. In Fig. 4-16(a), we see a thick pipe leading to a small-capacity receiver. As long as the sender does not send more water than the bucket can contain, no water will be lost. In Fig. 4-16(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network. If too much water comes in too fast, it will back up and some will be lost (in this case by overflowing the funnel).

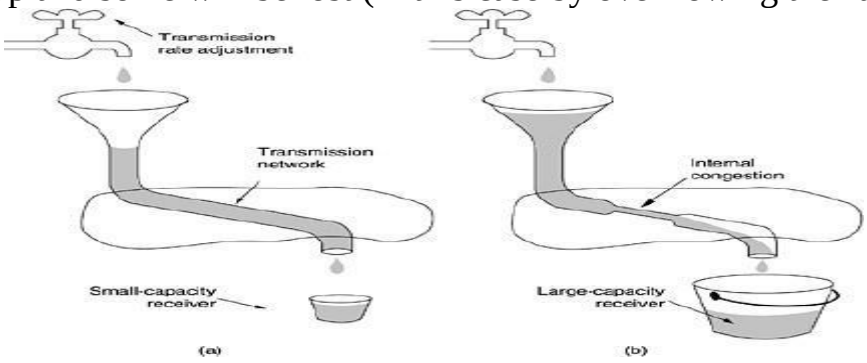


Figure 4-16. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

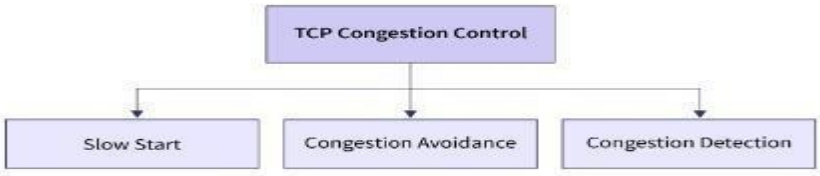


Figure 4-17. Approaches for Congestion Control

Congestion in TCP is handled by using these **three phases:**

1. Slow Start
2. Congestion Avoidance
3. Congestion Detection

Slow Start Phase

In the slow start phase, the sender sets congestion window size = maximum segment size (1 MSS) at the initial stage. The sender increases the size of the congestion window by 1 MSS after receiving the ACK (acknowledgment). The size of the congestion window increases exponentially in this phase. The **formula** for determining the size of the congestion window is Congestion window size = Congestion window size + Maximum segment size

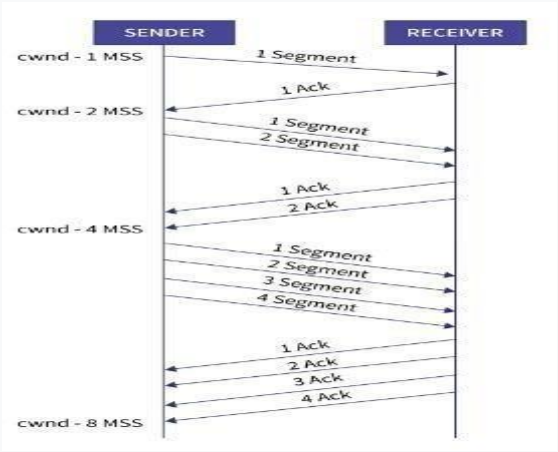


Figure 4-18. Window management in Congestion Control

Congestion Avoidance Phase

In this phase, after the threshold is reached, the size of the congestion window is increased by the sender linearly in order to avoid congestion. Each time an acknowledgment is received, the sender increments the size of the congestion window by 1.

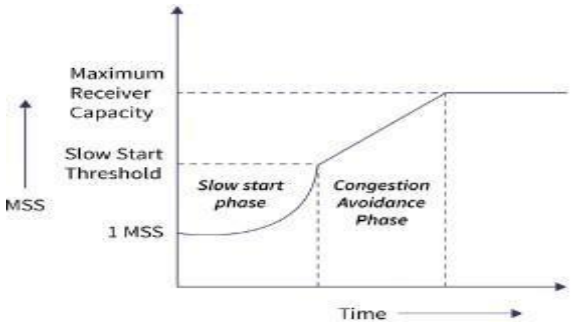


***Figure 4-19. Window management in Congestion* Avoidance**

Congestion Detection Phase

In this phase, the sender identifies the segment loss and gives acknowledgment depending on the type of loss detected.

UNIT-V

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

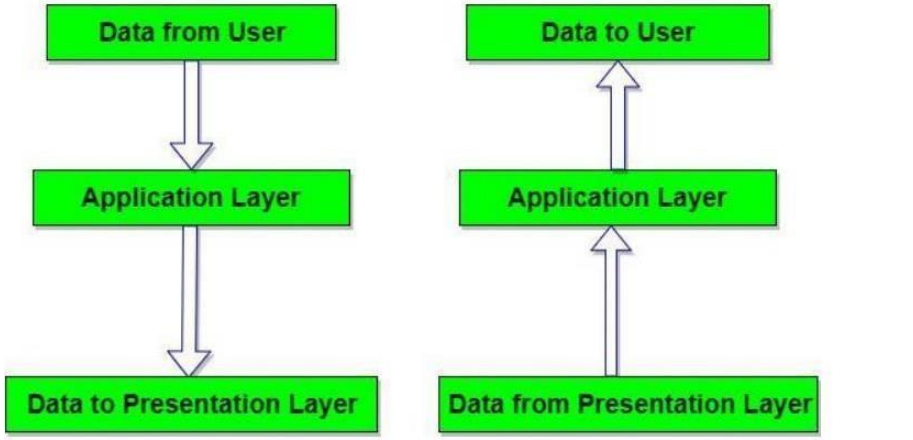
1. Mail Services: This layer provides the basis for E-mail forwarding and storage.

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 Computer Networks Page 113 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.

5. Addressing: To obtain communication between client and server, there is a need for addressing. When a client made a request to the server, the request contains the server address and its own address. The server response to the client request, the request contains the destination address, i.e., client address. To achieve this kind of addressing, DNS is used.


Data from User
Application Layer
Data to Presentation Layer
Data to User
Application Layer
Data from Presentation Layer

Application Layer Paradigms: -

Client-Server Model:-

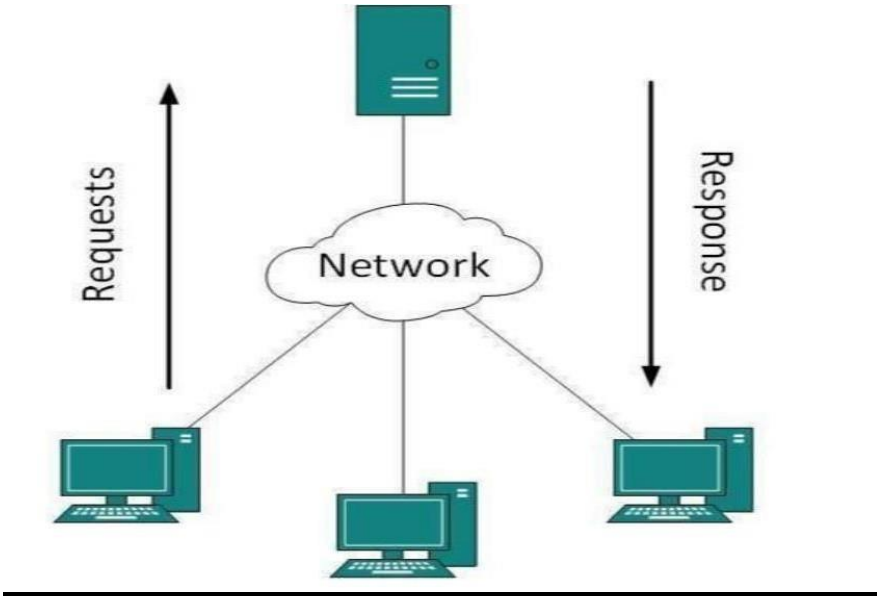
Two remote application processes can communicate mainly in two different fashions:

• 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.

In Client-server architecture, clients do not directly communicate with each other. For example, in a web application, two browsers do not directly communicate with each other.

A server is fixed, well-known address known as IP address because the server is always on while the client can always contact the server by sending a packet to the sender's IP address.



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)

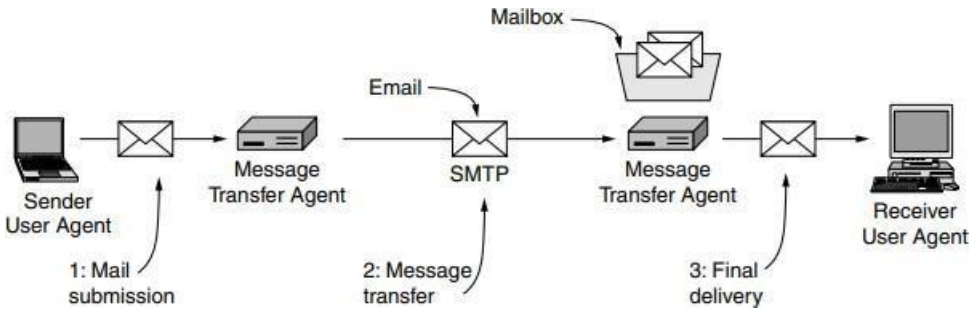
The Hyper Text Transfer Protocol (HTTP) is the foundation of World Wide Web. Hypertext is well organized documentation system which uses hyperlinks to link the pages in the text documents. HTTP works on client server model. When a user wants to access any HTTP page on the internet, the client machine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

Email

E-mail is defined as the transmission of messages on the Internet. It is one of the most commonly used features over communications networks that may contain text, files, images, or other attachments. Generally, it is information that is stored on a computer sent through a network to a specified individual or group of individuals.

Email messages are conveyed through email servers; it uses multiple protocols within the TCP/IP suite.

E-mail Protocols are set of rules that help the client to properly transmit the information to or from the mail server. Here in this tutorial, we will discuss various protocols such as **SMTP, POP,** and **IMAP.**



ARCHITECTURE OF EMAIL SYSTEM

SMTP

SMTP stands for **Simple Mail Transfer Protocol**. It was first proposed in 1982. It is a standard protocol used for sending e-mail efficiently and reliably over the internet.

Key Points:

- SMTP is application level protocol.
- SMTP is connection oriented protocol.
- SMTP is text based protocol.
- It handles exchange of messages between e-mail servers over TCP/IP network.
- Apart from transferring e-mail, SMPT also provides notification regarding incoming mail.
- When you send e-mail, your e-mail client sends it to your e-mail server which further contacts the recipient mail server using SMTP client.

- These SMTP commands specify the sender's and receiver's e-mail address, along with the message to be send.
- The exchange of commands between servers is carried out without intervention of any user.
- In case, message cannot be delivered, an error report is sent to the sender which makes SMTP a reliable protocol.

IMAP

IMAP stands for **Internet Message Access Protocol.** It was first proposed in 1986. There exist five versions of IMAP as follows:

1. Original IMAP
2. IMAP2
3. IMAP3
4. IMAP2bis
5. IMAP4

Key Points:

- IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.
- The e-mail is hold and maintained by the remote server.
- It enables us to take any action such as downloading, delete the mail without reading the mail.It enables us to create, manipulate and delete remote message folders called mail boxes.
- IMAP enables the users to search the e-mails.
- It allows concurrent access to multiple mailboxes on multiple mail servers.

POP

POP stands for Post Office Protocol. It is generally used to support a single client. There are several versions of POP but the POP 3 is the current standard.

Key Points

- POP is an application layer internet standard protocol.
- Since POP supports offline access to the messages, thus requires less internet usage time.
- POP does not allow search facility.
- In order to access the messaged, it is necessary to download them.
- It allows only one mailbox to be created on server.
- It is not suitable for accessing non mail data.
- POP commands are generally abbreviated into codes of three or four letters. Eg. STAT.

WWW

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with internet, enables the retrieval and display of text and media to your device.

The World Wide Web is based on several different technologies: Web browsers, Hypertext Markup Language (HTML) and Hypertext Transfer Protocol (HTTP).

A Web browser is used to access web pages. Web browsers can be defined as programs which display text, data, pictures, animation and video on the Internet. Hyperlinked resources on the World Wide Web can be accessed using software interfaces provided by Web browsers. Initially, Web browsers were used only for surfing the Web but now they have become more universal. Web browsers can be used for several tasks including conducting searches, mailing, transferring files, and much more. Some of the commonly used browsers are Internet Explorer, Opera Mini, and Google Chrome.

There are 3 components of the web:

Uniform Resource Locator (URL): serves as a system for resources on the web.

HyperText Transfer Protocol (HTTP): specifies communication of browser and server.

Hyper Text Markup Language (HTML): defines the structure, organisation and content of a webpage.

HTTP is based on the client/server idea, having a client and a server program, both of which can be executed on different end systems. The communication is carried out through an exchange of HTTP messages. This protocol specifies the structure of these messages. For example, HTTP defines how a pair of client/server hosts should exchange messages. In this context, a Web page consists of files, such as Hypertext Mark-up Language (HTML) file or an image that can be addressed by a single uniform resource locator (URL).

A URL is a global address of an HTML document and has two parts. The first part indicates what protocol is used, and the second part determines the IP address of the associated resource

Telnet:-

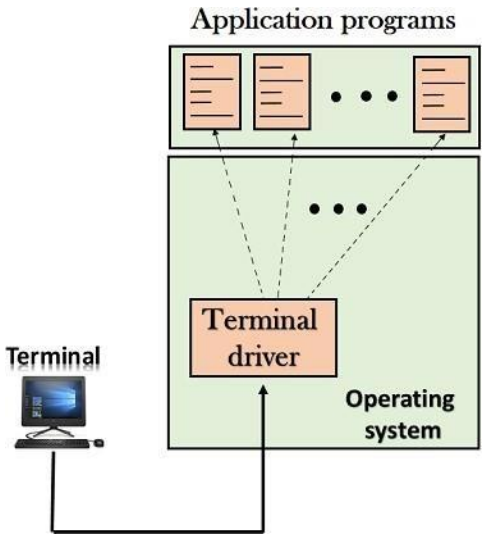
Telnet is a network protocol that allows a user to communicate with a remote device. It is a virtual terminal protocol used mostly by network administrators to remotely access and manage devices. Administrator can access the device by telnetting to the IP address or hostname of a remote device. To use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

- The main task of the internet is to provide services to users. For example, users want to run different application programs at the remote site and transfers a result to the local site. This requires a client-server program such as FTP, SMTP. But this would not allow us to create a specific program for each demand.
- The better solution is to provide a general client-server program that lets the user access any application program on a remote computer. Therefore, a program that allows a user to log on to a remote computer. A popular client-server program Telnet is used to meet such demands. Telnet is an abbreviation for **Terminal Network**.

- Telnet provides a connection to the remote computer in such a way that a local terminal appears to be at the remote side.

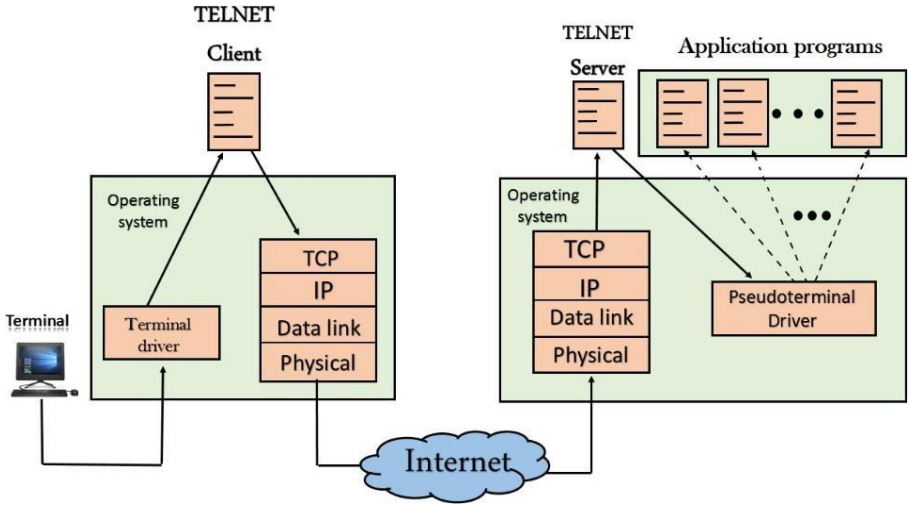
There are two types of login:

Local Login



- When a user logs into a local computer, then it is known as local login.
- When the workstation running terminal emulator, the keystrokes entered by the user are accepted by the terminal driver. The terminal driver then passes these characters to the operating system which in turn, invokes the desired application program.
- However, the operating system has special meaning to special characters. For example, in UNIX some combination of characters have special meanings such as control character with "z" means suspend. Such situations do not create any problem as the terminal driver knows the meaning of such characters. But, it can cause the problems in remote login.

Remote login



- When the user wants to access an application program on a remote computer, then the user must perform remote login.

How remote login occurs

At the local site

The user sends the keystrokes to the terminal driver, the characters are then sent to the TELNET client. The TELNET client which in turn, transforms the characters to a universal character set known as network virtual terminal characters and delivers them to the local TCP/IP stack

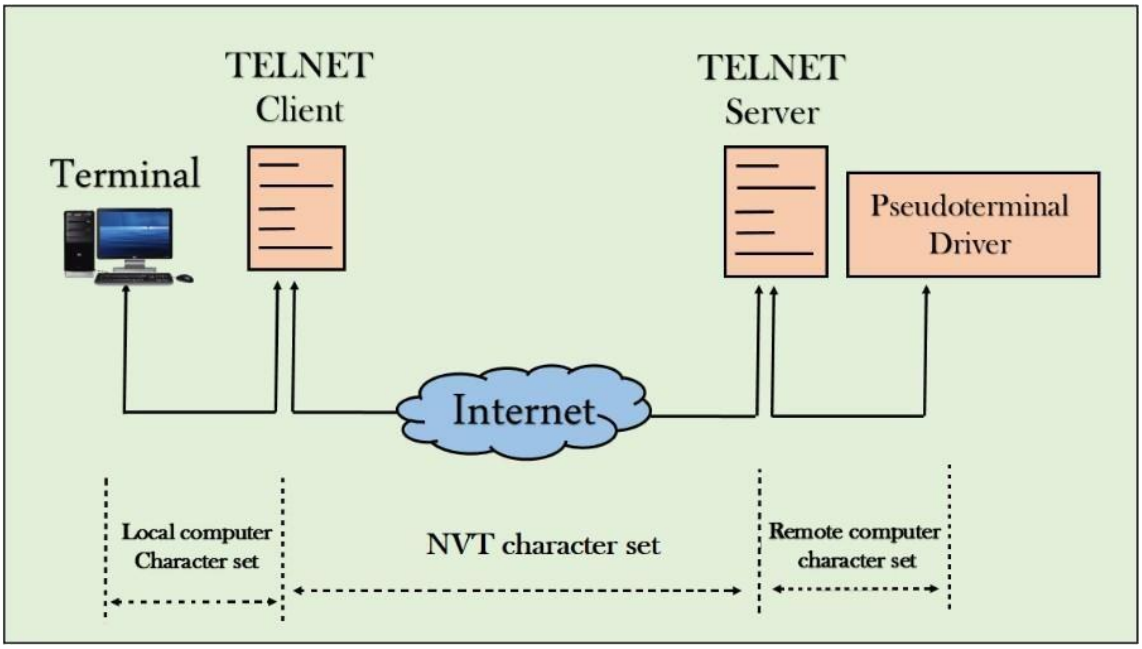
At the remote site

The commands in NVT forms are transmitted to the TCP/IP at the remote machine. Here, the characters are delivered to the operating system and then pass to the TELNET server. The TELNET server transforms the characters which can be understandable by a remote computer. However, the characters cannot be directly passed to the operating system as a remote operating system does not receive the characters from the TELNET server. Therefore it requires some piece of software that can accept the characters from the TELNET server. The operating system then passes these characters to the appropriate application program.

Network Virtual Terminal (NVT)

- The network virtual terminal is an interface that defines how data and commands are sent across the network.

- In today's world, systems are heterogeneous. For example, the operating system accepts a special combination of characters such as end-of-file token running a DOS operating system *ctrl+z* while the token running a UNIX operating system is *ctrl+d*.
- TELNET solves this issue by defining a universal interface known as network virtual interface.
- The TELNET client translates the characters that come from the local terminal into NVT form and then delivers them to the network. The Telnet server then translates the data from NVT form into a form which can be understandable by a remote computer.



DNS:-

DNS is a hostname for IP address translation service. DNS is a distributed database implemented in a hierarchy of name servers. It is an application layer protocol for message exchange between clients and servers.

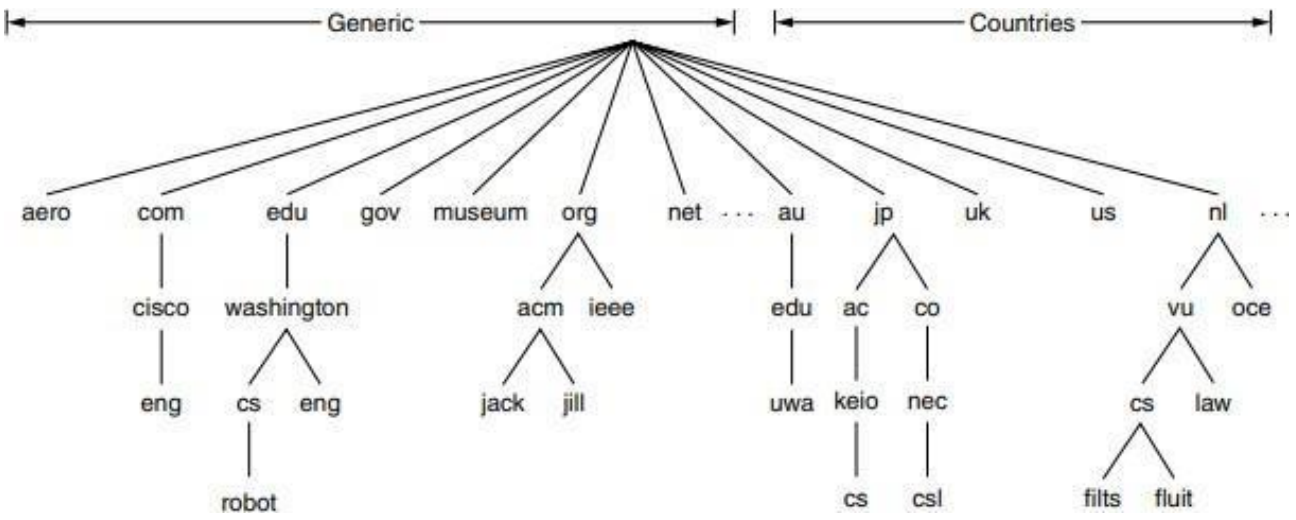
Requirement: Every host is identified by the IP address but remembering numbers is very difficult for the people also the IP addresses are not static therefore a mapping is required to change the domain name to the IP address. So DNS is used to convert the domain name of the websites to their numerical IP address.

Domain: There are various kinds of DOMAIN:

Generic domain: .com(commercial) .edu(educational) .mil(military) .org(non profit organization) .net(similar to commercial) all these are generic domain.

Country domain .in (india) .us .uk

Inverse domain if we want to know what is the domain name of the website. Ip to domain name mapping. So DNS can provide both the mapping



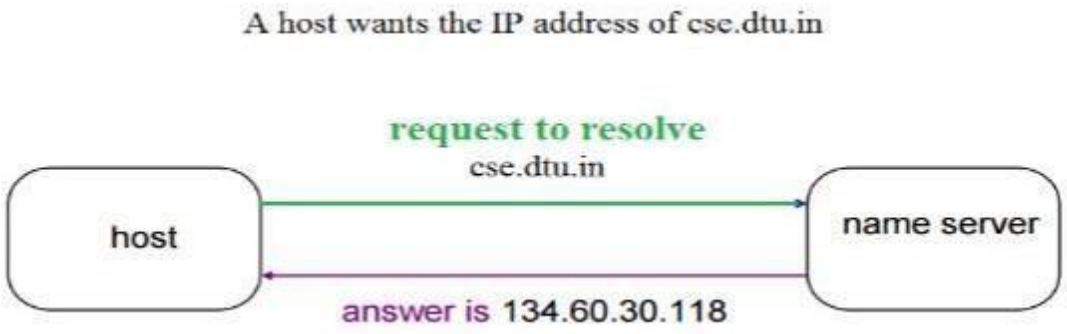
It is very difficult to find out the ip address associated to a website because there are millions of websites and with all those websites we should be able to generate the ip address immediately, there should not be a lot of delay for that to happen organization of database is very important.

DNS record: Domain name, ip address what is the validity?? what is the time to live ?? and all the information related to that domain name. These records are stored in tree like structure.

Namespace: Set of possible names, flat or hierarchical. The naming system maintains a collection of bindings of names to values – given a name, a resolution mechanism returns the corresponding value.

Name server: It is an implementation of the resolution mechanism. DNS (Domain Name System) = Name service in Internet – Zone is an administrative unit, domain is a subtree.

Name to Address
Resolution:



The host requests the DNS name server to resolve the domain name. And the name server returns the IP address corresponding to that domain name to the host so that the host can future connect to that IP address.

Hierarchy of Name Servers Root name servers: It is contacted by name servers that can not resolve the name. It contacts authoritative name server if name mapping is not known. It then gets the mapping and returns the IP address to the host.

Top level domain (TLD) server: It is responsible for com, org, edu etc and all top level country domains like uk, fr, ca, in etc. They have info about authoritative domain servers and know the names and IP addresses of each authoritative name server for the second-level domains.

Authoritative name servers are the organization's DNS server, providing authoritative hostName to IP mapping for organization servers. It can be maintained by an organization or service provider. In order to reach cse.dtu.in we have to ask the root DNS server, then it will point out to the top level domain server and then to authoritative domain name server which actually contains the IP address. So the authoritative domain server will return the associative ip address.

Domain Name Server

The client machine sends a request to the local name server, which , if root does not find the address in its database, sends a request to the root name server , which in turn, will route the query to an top-level domain (TLD) or authoritative name server. The root name server can also contain some hostName to IP address mappings. The Top-level domain (TLD) server always knows who the authoritative name server is. So finally the IP address is returned to the local name server which in turn returns the IP address to the host.