DATA COMMUNICATIONS AND NETWORKING

B.TECH
(IV YEAR – I SEM & III YEAR – I SEM)

Department of Electronics and Communication Engineering

SVR ENGINEERING COLLEGE
NANDYAL.

AYYALURU METTA, NANDYAL– 518 503 (A.P)

(Affiliated to JNTUA Anantapur, Approved by AICTE, New Delhi)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech IV-ISem. (ECE)

L	T	P	C
3	1	0	3

15A04704   DATA COMMUNICATIONS & NETWORKING

UNIT-I
Introduction to Networks & Data Communications
The Internet, Protocols & Standards, Layered Tasks, OSI Model, TCP / IP, Addressing, Line Coding Review, Transmission Media: Guided and unguided Media Review.

UNIT-II
Switching
Datagram Networks, Virtual Circuit Networks, Structure of a switch ,Ethernet Physical Layer, Data Link Layer: Error detection and Correction Data Link Control: Framing, Flow and Error Control Protocols, Noiseless Channel and Noisy Channel Protocol, HDLC, Point-to-Point Protocol.

UNIT-III
Multiple Access
RANDOH, CDMA, CSMA/CD, CSMA/CA, Controlled Access, Channelization, Wired LANs: IEEE Standards, Standard Ethernet, Fast Ethernet, Gigabit Ethernet, Wireless LAN, IEEE 802.11, Bluetooth IEEE 802.16.

UNIT-IV
Network Layer
Design Issues, Routing Algorithms, Congestion control, Algorithms.IPV4 Addresses, Connecting Devices, Virtual LAN IPV6 Addresses, Internet Protocol, Hardware Addressing versus IP Addressing, IP Data Gram.

UNIT-V
Transport Layer Protocol
 UDP and TCP, ATM, Cryptography, Network Security

Text Books:
1. B. A. Forouzan, "Data Communications and Networking", MGH, 4th ed. 2007.

Reference Books:
1. A. S. Tanenbaum, "Computer Networks", PHI.
2. W. Stallings, "Data and Computer Communication", PHI.

B.Tech (ECE) – III-I Sem	L	T	P	C
	3	0	0	3

(19A04504a)   DATA COMMUNICATIONS AND NETWORKING

Professional Elective-I

Course Objectives:

- To explain the basic concept of computer communication networks
- To demonstrate the TCP/IP and OSI models with merits and demerits.
- To explore the various layers of OSI Model.
- To introduce IP addressing, UDP and TCP Models.
- To have the concept of different routing techniques for data communications.

UNIT- I

Introduction to Computer Networks: Uses of computer Network, Network Software-design Issues for layers, Service primitives and relationship of services to Protocols, Reference models- OSI & TCP/IP, network architectures introduction, Example of Networks-X.25, Frame Relay & ATM, Protocols and Standards.

Learning Outcomes:

At the end of this unit, the student will be able to

- Enumerate the layers of the OSI model and TCP/IP. (L1)
- Explain the function(s) of each layer. (L2)

UNIT- II

Physical Layer: Physical layer - Data rate limits, Transmission media-guided and Unguided, Switching systems, Circuit switching, Datagram switching & Virtual circuit switching, Structure of circuit and packet switch, cable modem and DSL technologies, SONET basics, selection of IEEE std 802.11, a, b, c, g.

Learning Outcomes:

At the end of this unit, the student will be able to

- Understand principles of data communication using transmission (guided and wireless) media. (L1)
- Know to the concepts of various switching techniques. (L1)

- Explain the basics of DSL, SONET, and IEEE standards. (L2)

UNIT- III

Data link layer: Framing, Flow & Error control Protocols, HDLC, PPP, Multiple access techniques, random access, controlled access & Channelization, Ethernet types-bridged, Switched, Full duplex, Fast & gigabit Ethernet, Introduction to Data link layer in 802.11 LAN, connecting devices like passive hubs, repeaters, Active hubs, Bridges, Two-layer Switches, Routers, three layer switches, Gateway etc., Backbone networks, Virtual LANs, Simple Router architecture, Sliding window protocol.

Learning Outcomes:

At the end of this unit, the student will be able to

- List the different connecting devices for networking. (L1)
- Understand the principles of error control protocols, multiple access protocols, routers and switches in data link layer. (L1)
- Solve the error control and multiple access based problems. (L2)

UNIT- IV

Network Layer: IPv4 address, IPv6 address, Address mapping-ARP, RARP & DHCP, IPv4 datagram detail format, IPv6 datagram detail format, ICMP, IGMP, Network layer issues like Delivery, forwarding, intra-domain and Inter-domain routing, Routing algorithms like Shortest path routing, Flooding, Distance Vector Routing, Link State Routing, Path vector routing etc., Addressing types-Physical, Logical & port address.
Transport Layer: Transport layer-Process to process delivery, Connection oriented & Connectionless Transport, UDP, TCP, congestion control and Quality of Service.

Learning Outcomes:

At the end of this unit, the student will be able to

- Understand the concepts of IPvx and different protocols.(L1)
- Apply the knowledge on different routing algorithms and measure their performance metrics.(L2)
- Distinguish between the connection oriented and connection less transport protocols.(L4)

UNIT- V

Application Layer: Application layer protocols and applications like Ping, FTP, telnet, HTTP, SMTP, SNMP, TFTP, BOOTP, DNS, NFS, RPC, X-server, E-mail, Introduction to streaming Audio/Video,P2P file sharing, Introduction to socket programming.

Learning Outcomes:

At the end of this unit, the student will be able to

- Understand the importance of application layer and the terminology like FTP, HTTP, SMTP, SNMP,TFTP etc., (L1)
- Know about the P2P file sharing and socket programming.(L2)

Course Outcomes:

- Understand the requirement of theoretical & practical aspects of computer networks, functions of various layers involved in data communications, building the skills of sub netting and routing mechanisms.
- Explain the role of protocols in networking.
- Analyze the services and features of the various layers in the protocol stack.

TEXT BOOKS:

1. Behrouz A. Forouzan, "Data Communications and Networking", 4th Edition, Tata McGraw Hill, 2007.
2. Andrew Tenenbaum, "Computer Networks", 4th Edition, Pearson Education.
3. Kurose & Ross, "Computer Networking- A top down approach featuring the Internet", 3rd Edition, Pearson Education.
4. William Stallings, "Computer Networks and Cryptography", 3rd Edition, Pearson Education.

REFERENCES:

1. Behrouz A. Forouzan, "TCP/IP protocol Suit", 3rd Edition, Tata McGraw Hill Publications.
2. Stevens, "TCP/IP illustrated Volume - I & II", Pearson education.
3. Feibel Werner, "Encyclopedia of networking", Pearson education.

<center>**Unit-I**</center>

<center>**Introduction to Networks &Data Communications**</center>

In Data Communications, <u>*data*</u> generally are defined as information that is stored in digital form. <u>*Data communications*</u> is the process of transferring digital information between two or more points. <u>*Information*</u> is defined as the knowledge or intelligence. Data communications can be summarized as the transmission, reception, and processing of digital information. For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs). The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

A data communications system has five components:

1. **Message:** The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.

2. **Sender:** The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.

3. **Receiver:** The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.

4. **Transmission medium:** The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.

5. **Protocol**: A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices.

<u>The Internet:</u>

The **Advanced Research Projects Agency Network** (**ARPANET**) was an early packet switching network and the first network to implement the protocol suite TCP/IP. Both technologies became the technical foundation of the Internet. The ARPANET was initially funded by the Advanced Research Projects Agency (ARPA) of the United States Department of Defense. The packet switching methodology employed in the ARPANET was based on concepts and designs by Americans Leonard Kleinrock and Paul Baran, British scientist Donald Davies, and Lawrence Roberts. The TCP/IP communications protocols were developed for the ARPANET

by computer scientists Robert Kahn and Vint Cerf, and incorporated concepts from the French CYCLADES project directed by Louis Pouzin.

As the project progressed, protocols for internetworking were developed by which multiple separate networks could be joined into a network of networks. Access to the ARPANET was expanded in 1981 when the National Science Foundation (NSF) funded the Computer Science Network (CSNET). In 1982, the Internet protocol suite(TCP/IP) was introduced as the standard networking protocol on the ARPANET. In the early 1980s the NSF funded the establishment for national supercomputing centers at several universities, and provided interconnectivity in 1986 with the NSFNET project, which also created network access to the supercomputer sites in the United States from research and education organizations. The ARPANET was decommissioned in 1990.

PROTOCOLS & Standards

Protocol: A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices.

- Syntax
- Semantics
- Timing

An association of organizations, governments, manufacturers and users form the standards organizations and are responsible for developing, coordinating and maintaining the standards. The intent is that all data communications equipment manufacturers and users comply with these standards. The primary standards organizations for data communication are:

1. **International Standard Organization (ISO)**

   ISO is the international organization for standardization on a wide range of subjects. It is comprised mainly of members from the standards committee of various governments throughout the world. It is even responsible for developing models which provides high level of system compatibility, quality enhancement, improved productivity and reduced costs. The ISO is also responsible for endorsing and coordinating the work of the other standards organizations.

2. **International Telecommunications Union-Telecommunication Sector(ITU-T)**

   ITU-T is one of the four permanent parts of the International Telecommunications Union based in Geneva, Switzerland. It has developed three sets of specifications: the *V series* for modem interfacing and data transmission over telephone lines, the *X series* for data transmission over public digital networks, email and directory services; the *I and Q series* for Integrated Services Digital Network (ISDN) and its extension Broadband ISDN. ITU-T membership consists of government authorities and representatives from many countries and it is the present standards organization for the United Nations.

3. **Institute of Electrical and Electronics Engineers (IEEE)**

   IEEE is an international professional organization founded in United States and is compromised of electronics, computer and communications engineers. It is currently the world's largest professional society with over 200,000 members. It develops communication and information processing standards with the underlying goal of advancing theory, creativity, and product quality in any field related to electrical engineering.

4. **American National Standards Institute (ANSI)**

   ANSI is the official standards agency for the United States and is the U.S voting representative for the ISO. ANSI is a completely private, non-profit organization comprised of equipment manufacturers and users of data processing equipment and services. ANSI membership is comprised of people form professional societies, industry associations, governmental and regulatory bodies, and consumer goods.

5. **Electronics Industry Association (EIA)**

   EIA is a non-profit U.S. trade association that establishes and recommends industrial standards. EIA activities include standards development, increasing public awareness, and lobbying and it is responsible for developing the RS (recommended standard) series of standards for data and communications.

Layered Tasks

To reduce the design complexity, most of the networks are organized as a series of **layers** or **levels**, each one build upon one below it. The basic idea of a layered architecture is *to divide the design into small pieces*. Each layer adds to the services provided by the lower layers in such a manner that the highest layer is provided a full set of services to manage communications and run the applications. The benefits of the layered models are modularity and clear interfaces, i.e. open architecture and comparability between the different providers' components. A basic principle is to ensure independence of layers by defining services provided by each layer to the next higher layer without defining how the services are to be performed. This permits changes in a layer without affecting other layers. The basic elements of a layered model are services, protocols and interfaces. A *service* is a set of actions that a layer offers to another (higher) layer. *Protocol* is a set of rules that a layer uses to exchange information with a peer entity. These rules concern both the contents and the order of the messages used. Between the layers service interfaces are defined. The messages from one layer to another are sent through those interfaces.
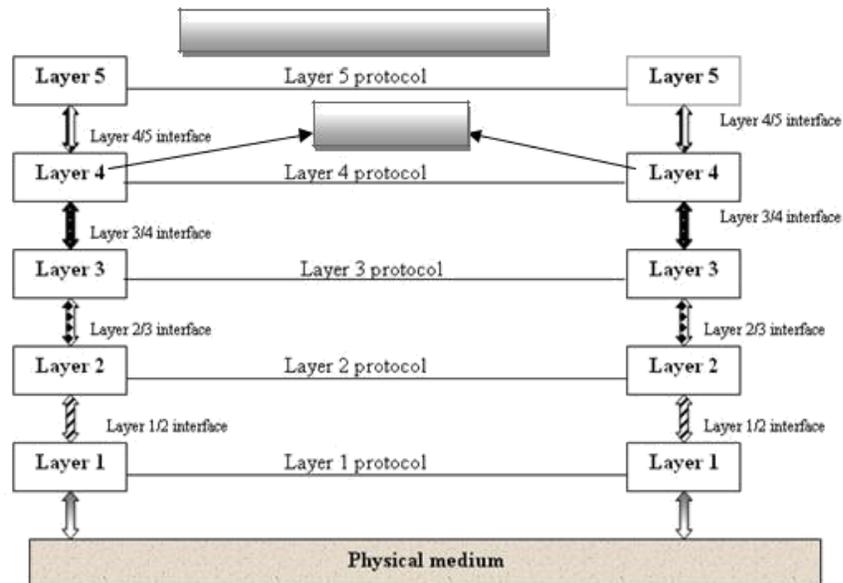
Figure: Layered Architecture

In a *n-layer* architecture, layer n on one machine carries on conversation with the layer n on other machine. The rules and conventions used in this

conversation are collectively known as the *layer-n protocol*. Basically, a protocol is an agreement between the communicating parties on how communication is to proceed. Five-layer architecture is shown below**;** the entities comprising the corresponding layers on different machines are called ***peers***. In other words, it is the peers that communicate using protocols. In reality, no data is transferred from layer n on one machine to layer n of another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached. Below layer-1 is the physical layer through which actual communication occurs.

 With layered architectures, communications between two corresponding layers requires a unit of data called a ***protocol data unit (PDU)***. A PDU can be a header added at the beginning of a message or a trailer appended to the end of a message. Data flows downward through the layers in the source system and upwards at the destination address. As data passes from one layer into another, headers and trailers are added and removed from the PDU. This process of adding or removing PDU information is called ***encapsulation/decapsulation.*** Between each pair of adjacent layers there is an ***interface***. The *interface* defines which primitives operations and services the lower layer offers to the upper layer adjacent to it. A set of layers and protocols is known as **network architecture**. A list of protocols used by a certain system, one protocol per layer, is called **protocol stack**.

OSI MODEL

The OSI model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983). It was revised in 1995(Day, 1995). The model is called the ISO-OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems.The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems. It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network.

7|	Application
6|	Presentation
5|	Session
4|	Transport
3|	Network
2|	Data link
1		Physical
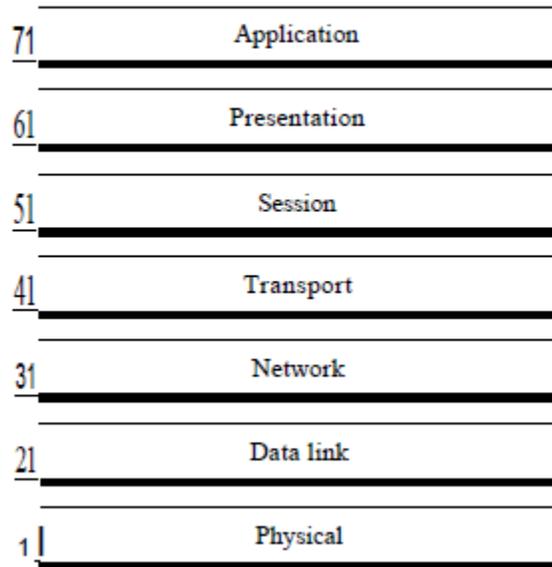
The OSI model is composed of seven ordered layers: physical (layer 1), data link (layer 2), network (layer 3), transport (layer 4), session (layer 5), presentation (layer 6), and application (layer 7). Figure below shows the layers involved when a message is sent from device A to device B. As the message travels from A to B, it may pass through many intermediate nodes. These intermediate nodes usually involve only the first three layers of the OSI model.

1. Physical Layer

The physical layer coordinates the functions required to carry a bit stream over a physical medium. It deals with the mechanical and electrical specifications of the interface and transmission medium. It also defines the procedures and functions that physical devices and interfaces have to perform for transmission to occur.

Layer						Name of unit exchanged
7	Application		Application protocol		Application	APDU
Interface						
6	Presentation		Presentation protocol		Presentation	PPDU
5	Session		Session protocol		Session	SPDU
4	Transport		Transport protocol		Transport	TPDU

Communication subnet boundary

Internal subnet protocol

3	Network	Network	Network	Network	Packet
2	Data link	Data link	Data link	Data link	Frame
1	Physical	Physical	Physical	Physical	Bit
	Host A	Router	Router	Host B	

— Network layer host-router protocol
— Data link layer host-router protocol
— Physical layer host-router protocol

The physical layer is also concerned with the following:

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

o **Representation of bits**. The physical layer data consists of a stream of bits (sequence of Os or 1s) with no interpretation. To be transmitted, bits must be encoded into signals--electrical or optical. The physical layer defines the type of encoding .

o **Data rate.** The transmission rate-the number of bits sent each second-is also defined by the physical layer. In other words, the physical layer defines the duration of a bit, which is how long it lasts.

o **Synchronization of bits**. The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.

o **Line configuration**. The physical layer is concerned with the connection of devices to the

media. In a point-to-point configuration, two devices are connected through a dedicated link. In a multipoint configuration, a link is shared among several devices.

o **Physical topology**. The physical topology defines how devices are connected to make a network. Devices can be connected by using a mesh topology (every device is connected to every other device), a star topology (devices are connected through a central device), a ring topology (each device is connected to the next, forming a ring), a bus topology (every device is on a common link), or a hybrid topology (this is a combination of two or more topologies).

o **Transmission mode**. The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex. In simplex mode, only one device can send; the other can only receive. The simplex mode is a one-way communication. In the half-duplex mode, two devices can send and receive, but not at the same time. In a full-duplex (or simply duplex) mode, two devices can send and receive at the same time.

2. Data Link Layer

The data link layer transforms the physical layer, a raw transmission facility, to a reliable link. It makes the physical layer appear error-free to the upper layer (network layer).

Other responsibilities of the data link layer include the following:

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

o **Physical addressing**. If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects the network to the next one.

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

o **Error control**. The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplicate frames. Error control is normally achieved through a trailer added to the end of the frame.

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

3. Network Layer

The network layer is responsible for the source-to-destination delivery of a packet, possibly

across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination. If two systems are connected to the same link, there is usually no need for a network layer. However, if the two systems are attached to different networks (links) with connecting devices between the networks (links), there is often a need for the network layer to accomplish source-to-destination delivery.

Other responsibilities of the network layer include the following:

o **Logical addressing**. The physical addressing implemented by the data link layer handles the addressing problem locally. If a packet passes the network boundary, we need another addressing system to help distinguish the source and destination systems. The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver.

o **Routing**. When independent networks or links are connected to create *intemetworks* (network of networks) or a large network, the connecting devices (called *routers* or *switches)* route or switch the packets to their final destination. One of the functions of the network layer is to provide this mechanism.

4. Transport Layer

The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on a host. Whereas the network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.

Other responsibilities of the transport layer include the following:

o **Service-point addressing**. Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other. The transport layer header must therefore include a type of address called a *service-point address* (or port address). The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on that computer.

o **Segmentation and reassembly**. A message is divided into transmittable segments, with each

segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.

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

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

o **Error control.** Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed process-to-process rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication). Error correction is usually achieved through retransmission.

5. Session Layer

The services provided by the first three layers (physical, data link, and network) are not sufficient for some processes. The session layer is the network *dialog controller.* It establishes, maintains, and synchronizes the interaction among communicating systems.

Specific responsibilities of the session layer include the following:

o **Dialog control.** The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half duplex (one way at a time) or full-duplex (two ways at a time) mode.

o **Synchronization**. The session layer allows a process to add checkpoints, or synchronization points, to a stream of data.

6. Presentation Layer

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

Specific responsibilities of the presentation layer include the following:

o **Translation**. The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. The information must be changed to bit streams before being transmitted. Because different computers use different

encoding systems, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the information from its sender-dependent format into a common format. The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

o **Encryption**. To carry sensitive information, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.

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

7. Application Layer

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

Specific services provided by the application layer include the following:

o **Network virtual terminal**. A network virtual terminal is a software version of a physical terminal, and it allows a user to log on to a remote host.

o **File transfer, access, and management**. This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.

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

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

TCP/IP PROTOCOL SUITE

The TCPIIP protocol suite was developed prior to the OSI model. Therefore, the layers in the TCP/IP protocol suite do not exactly match those in the OSI model. The original TCP/IP protocol suite was defined as having four layers: host-to-network, internet, transport, and application. However, when TCP/IP is compared to OSI, we can say that the host-to-network layer is equivalent to the combination of the physical and data link layers. The internet layer is equivalent to the network layer, and the application layer is roughly doing the job of the session, presentation, and application layers with the transport layer in TCP/IP taking care of part of the

duties of the session layer.

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

At the transport layer, *TCP/IP* defines three protocols: Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP). At the network layer, the main protocol defined by TCP/IP is the Internetworking Protocol (IP); there are also some other protocols that support data movement in this layer.
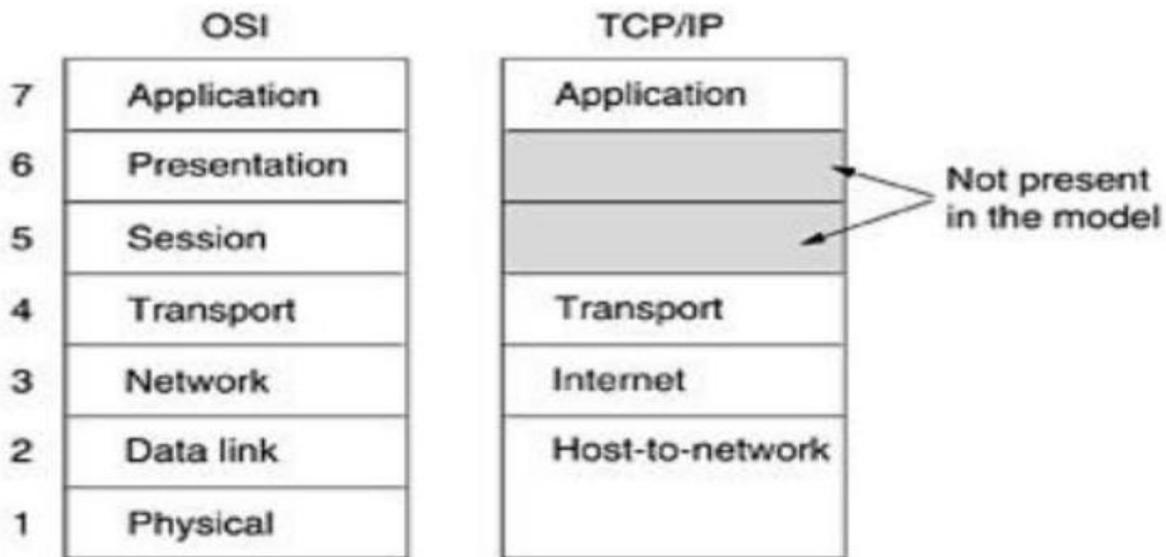
1. Host-to-Network Layer:

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.

2. Internet Layer:

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.

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.

3. 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.

4. The Application Layer:

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

Comparison of the OSI and TCP/IP Reference Models:

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 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.

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.

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.

Addressing:

A protocol such as IPv4 that defines addresses has an address space. An address space is the total number of addresses used by the protocol. If a protocol uses N bits to define an address, the address space is 2^N because each bit can have two different values (0 or 1) and N bits can have 2^N values. IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than 4 billion). This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet.

Line Coding Review

Line coding

For reliable clock recovery at the receiver, one usually imposes a maximum run length constraint on the generated channel sequence, i.e., the maximum number of consecutive ones or zeros is bounded to a reasonable number. A clock period is recovered by observing transitions in the received sequence, so that a maximum run length guarantees such clock recovery, while sequences without such a constraint could seriously hamper the detection quality.

After line coding, the signal is put through a physical communication channel, either a transmission medium or data storage medium.[2][3] Sometimes the characteristics of two very different-seeming channels are similar enough that the same line code is used for them. The most common physical channels are:

- The line-coded signal can directly be put on a transmission line, in the form of variations of the voltage or current (often using differential signaling).
- The line-coded signal (the "baseband" signal) undergoes further pulse shaping (to reduce its frequency bandwidth) and then modulated (to shift its frequency) to create an "RF signal" that can be sent through free space.
- The line-coded signal can be used to turn on and off a light source in free-space optical communication,   most commonly used in an infrared remote control.

The line-coded signal can be printed on paper to create a bar code.

The line-coded signal can be converted to magnetized spots on a hard drive or tape drive.

The line-coded signal can be converted to pits on an <u>optical disc</u>.
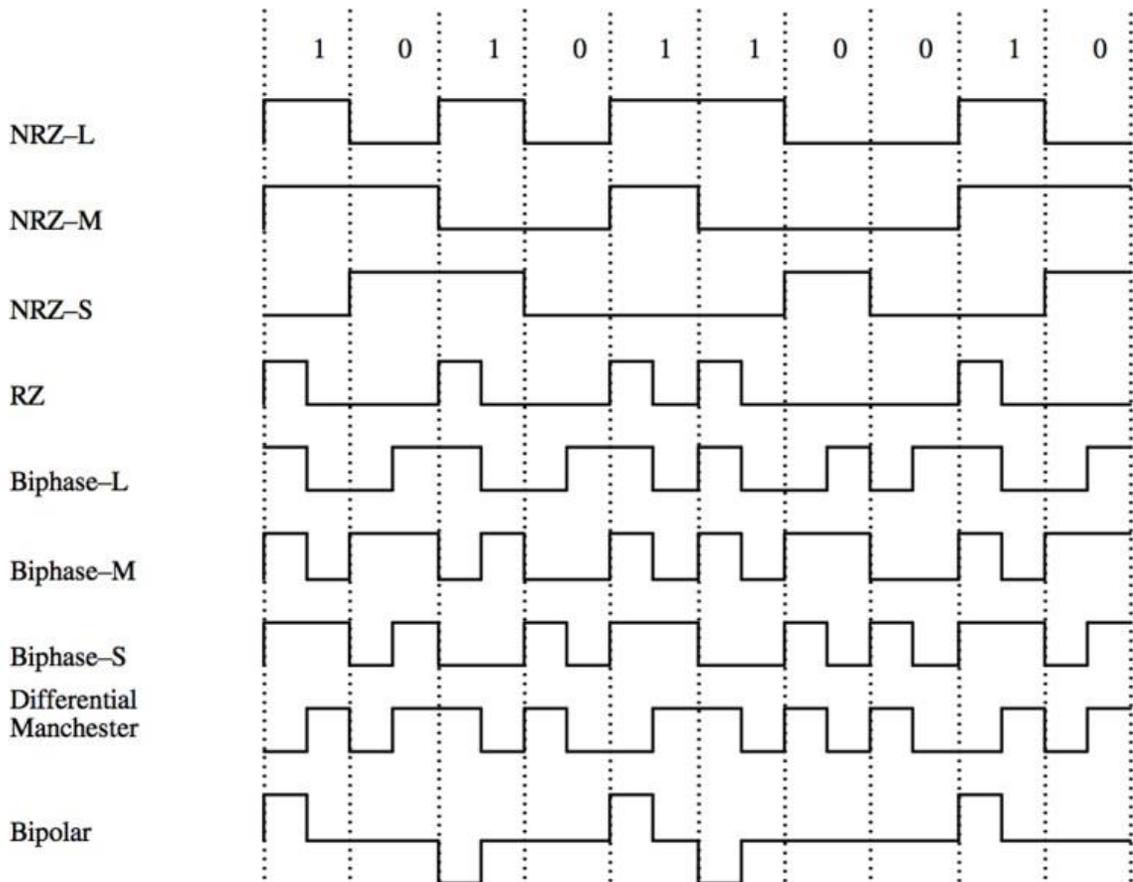
TRANSMISSION MEDIA

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

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

Guided Media

Guided media, which are those that provide a conduit from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial

cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light.

1. Twisted-Pair Cable

A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together, as shown in Figure 7.3.

Figure 7.3 *Twisted-pair cable*
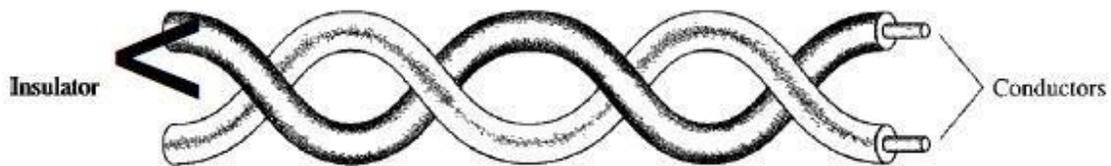
One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals. 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 lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables. Local-area networks, such as lOBase-T and lOOBase-T, also use twisted-pair cables.

2. Coaxial Cable

Coaxial cable (or *coax)* carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover (see Figure 7.7).

Figure 7.7   *Coaxial cable*
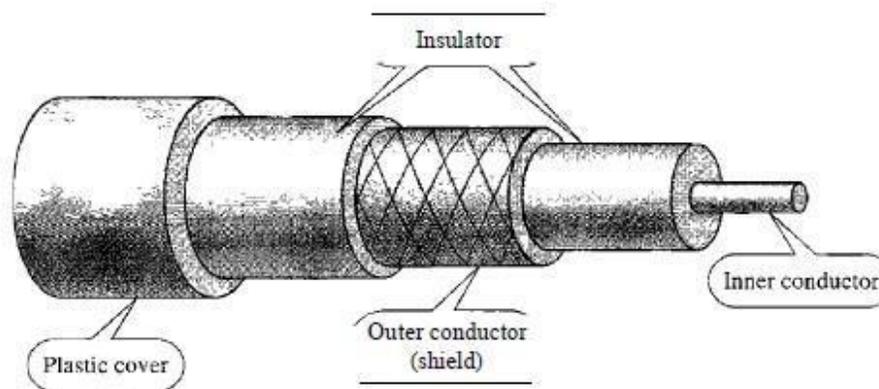
Applications

Coaxial cable was widely used in analog telephone networks where a single coaxial network could carry 10,000 voice signals. Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps. However, coaxial cable in telephone networks has largely been replaced today with fiber-optic cable. Cable TV networks also use coaxial cables. In the traditional cable TV network, the entire network used coaxial cable. Later, however, cable TV providers replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable. Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs.

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

Figure 7.10   *Bending of light ray*
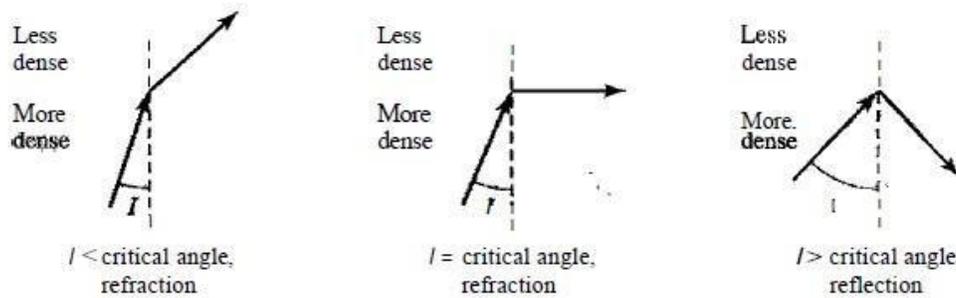
As the figure shows, if the angle of incidence *I* (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the critical angle, the ray refracts and moves closer to the surface. If the angle of incidence is equal to the critical angle, the light bends along the interface. If the angle is greater than the critical angle, the ray reflects (makes a turn) and travels again in the denser substance. Note that the critical angle is a property of the substance, and its value differs from one substance to another.

Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. See Figure 7.11.

Figure 7.11   *Optical jiber*
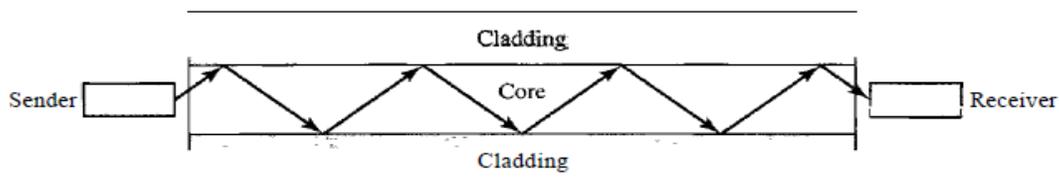
Figure 7.14 shows the composition of a typical fiber-optic cable. The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable. Kevlar is a strong material used in the fabrication of bulletproof vests. Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.

Figure 7.14 *Fiber construction*
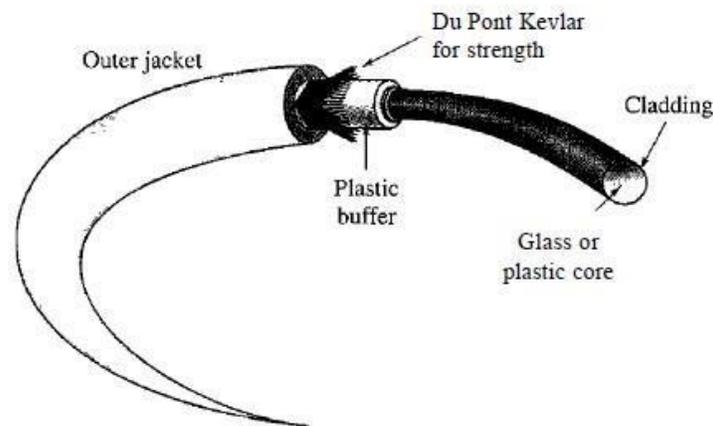
Applications

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).

a. **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.

b. **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.

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

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

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

f. **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.

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

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

c. **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-or-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

Figure 7.18 *Propagation methods*

Ground propagation
(below 2 MHz)

Sky propagation
(2-30 MHz)

Line-af-sight propagation
(above 30 MHz)
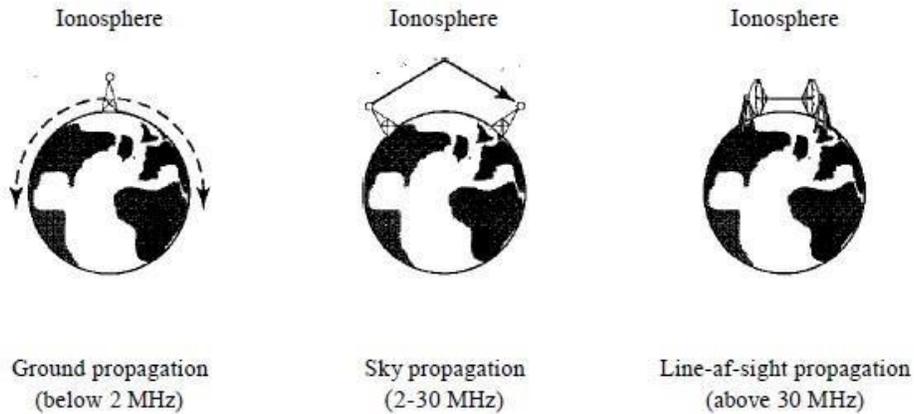
cannot be completely focused.

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.

UNIT II

Switching

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.

Figure 8.7 *A datagram network with four switches (routers)*
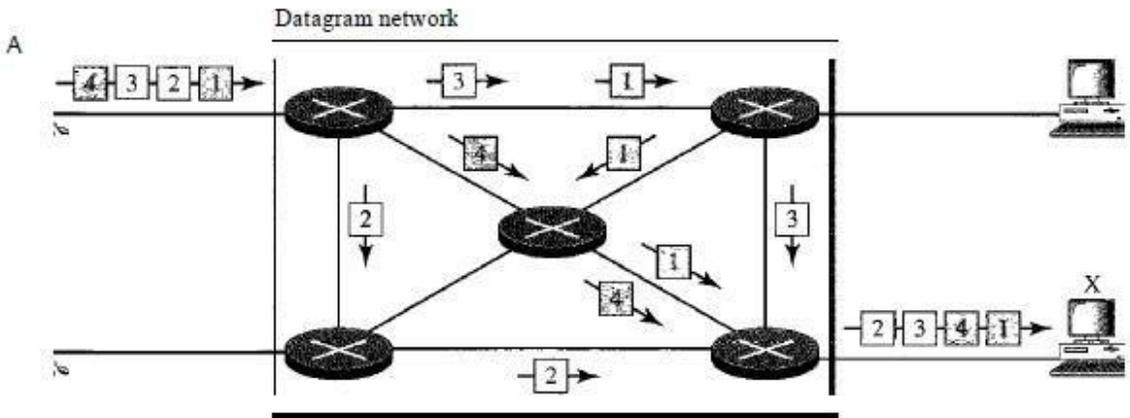
In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the

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 upper-layer 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.

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.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.

2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.

3. 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.

4. As in a circuit-switched network, all packets follow the same path established during the connection.

5. 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 8.10 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.

Figure 8.10   *Virtual-circuit network*
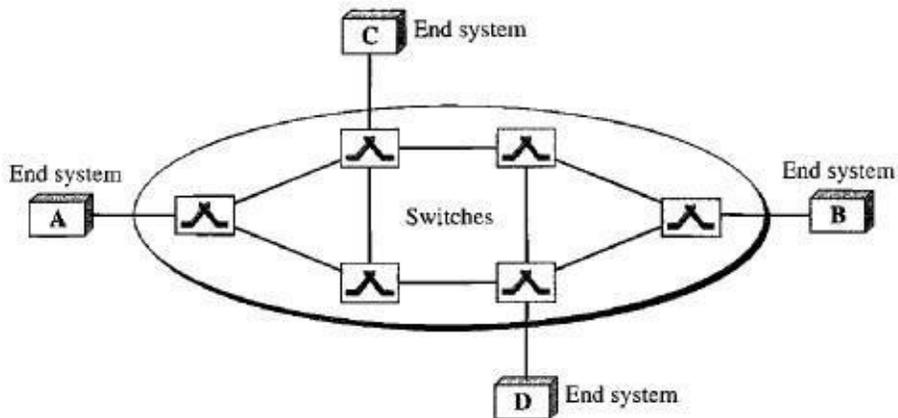
Structure of a switch

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 of the 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.

Figure 8.1 *Switched network*
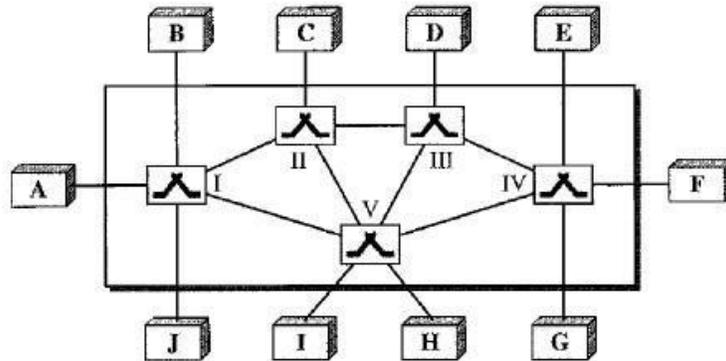
The end systems (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links. Traditionally, three methods of switching have been important: circuit switching, packet switching, and message switching. The first 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.

Ethernet Physical Layer

Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a 6-byte physical address. As shown in Figure 13.6, the Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.

Figure 13.6 *Example of an Ethernet address in hexadecimal notation*

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

6 bytes = 12 hex digits = 48 bits

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

or broadcast. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast. A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one-to-one. A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one-to-many. The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty-eight 1's.

For Example:

a)- 4A:30:10:21:1O:1A - This is a unicast address because A in binary is 1010 (even).

b)- 47:20:1B:2E:08:EE - This is a multicast address because 7 in binary is 0111 (odd).

c)- FF:FF:FF:FF:FF:FF - This is a broadcast address because all digits are F's.

The way the addresses are sent out on line is different from the way they are written in hexadecimal notation. The transmission is left-to-right, byte by byte; however, for each byte, the least significant bit is sent first and the most significant bit is sent last. This means that the bit that defines an address as unicast or multicast arrives first at the receiver.

Data Link Layer: Error detection and correction Data Link Control:

CYCLIC REDUNDANCY CHECK (CRC):

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word a_o to a_6' and the bits in the second word b_o to b_6.

Figure 10.14 *CRC encoder and decoder*
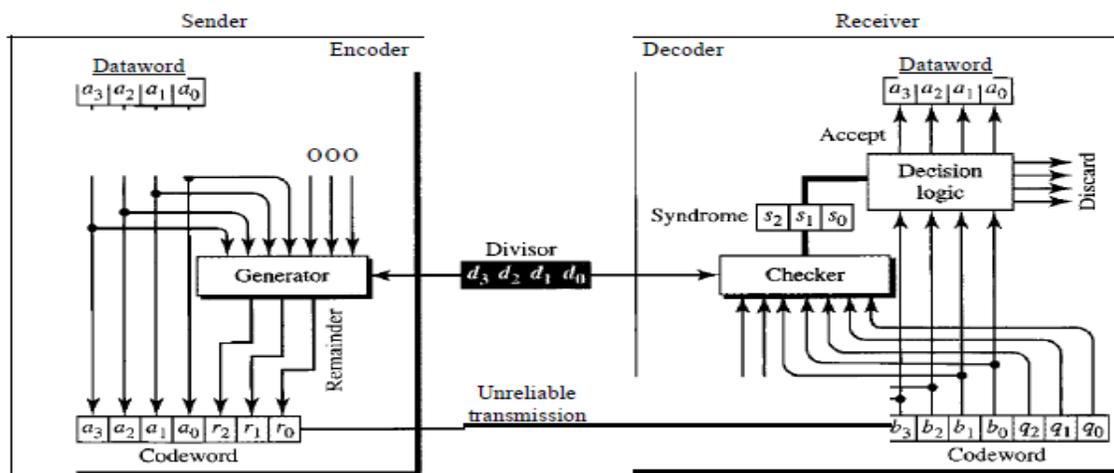
In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* - *k* (3 here) Os to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size *n* - *k* + I (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder *(r2r1r0)* is appended to the dataword to create the codeword. The decoder receives the possibly corrupted codeword. A copy of all *n* bits is fed to the checker which is a replica of the generator. The remainder produced by the checker is a syndrome of *n* - *k* (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes the dataword and augments it with *n* - *k* number of 0's. It then divides the augmented dataword by the divisor, as shown in Figure 10.15.

Figure 10.15 *Division in CRC encoder*
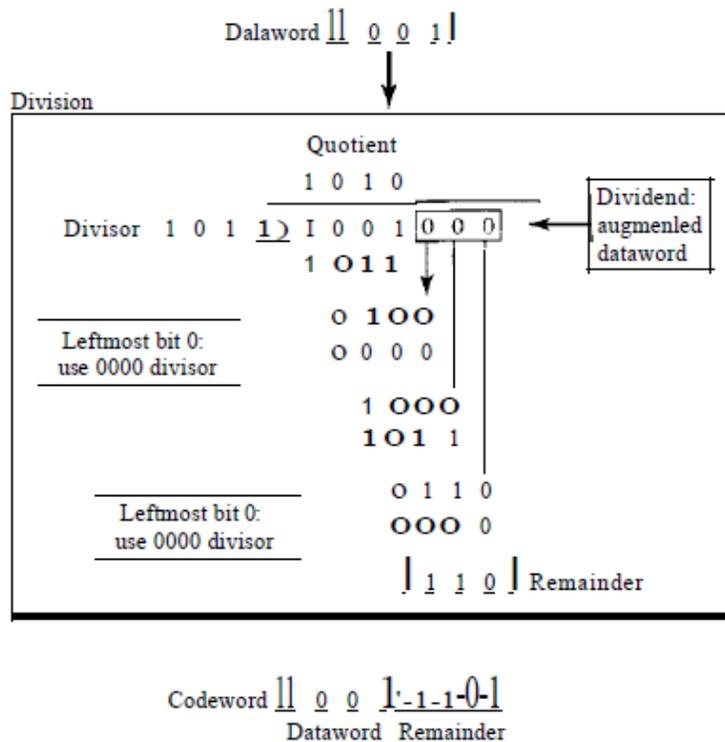
As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-Os divisor. When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits *(r2' r1* and *r0).* They are appended to the dataword to create the codeword.

Framing, Flow and Error Control Protocols

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all Os, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.16 shows two cases: The left hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all 0s .

Figure 10.7 *Division in the CRC decoder for two cases*
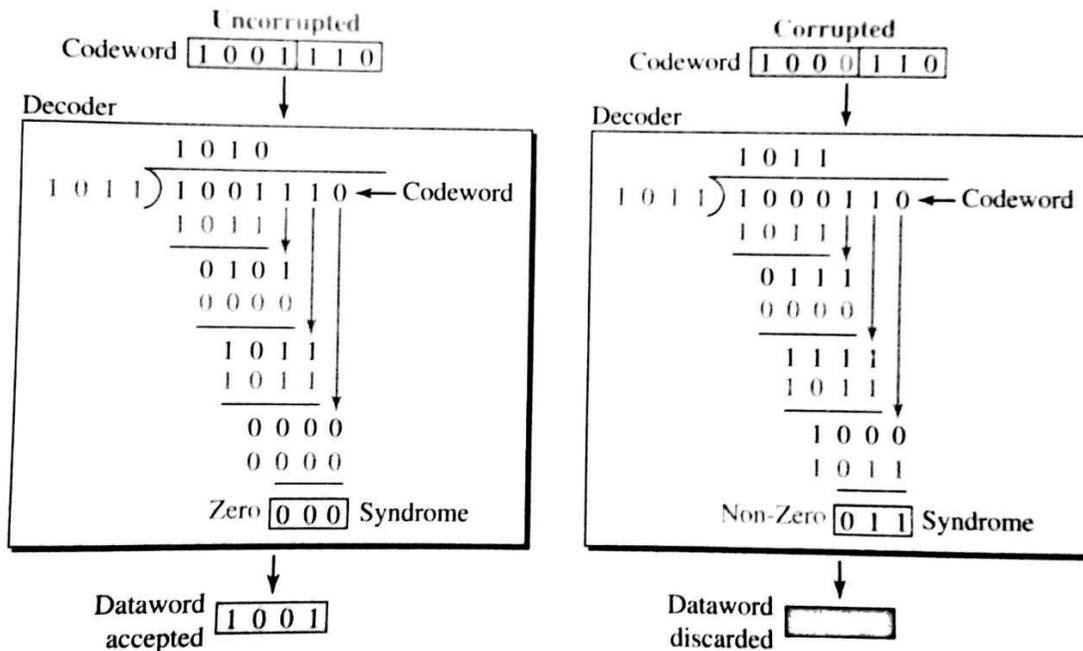
Divisor

Let us first consider the divisor. We need to note the following points:

1. The divisor is repeatedly XORed with part of the dividend.

2. The divisor has $n - k + 1$ bits which either are predefined or are all Os. In other words, the bits do not change from one dataword to another. In our previous example, the divisor bits were either 1011 or 0000. The choice was based on the leftmost bit of the part of the augmented data bits that are active in the XOR operation.

3. A close look shows that only $n - k$ bits of the divisor is needed in the XOR operation. The leftmost bit is not needed because the result of the operation is always 0, no matter what the value of this bit. The reason is that the inputs to this XOR operation are either both Os or both 1s. In our previous example, only 3 bits, not 4, is actually used in the XOR operation.

Polynomials

A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials.

A pattern of Os and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Figure 10.21 shows a binary pattern and its polynomial representation. In Figure 10.21a we show how to translate a binary pattern to a polynomial; in Figure 1O.21b we show how the polynomial can be shortened by removing all terms with zero coefficients.

Figure 10.21   *A polynomial to represent a binary word*

a. Binary pattern and polynomial

b. Short form
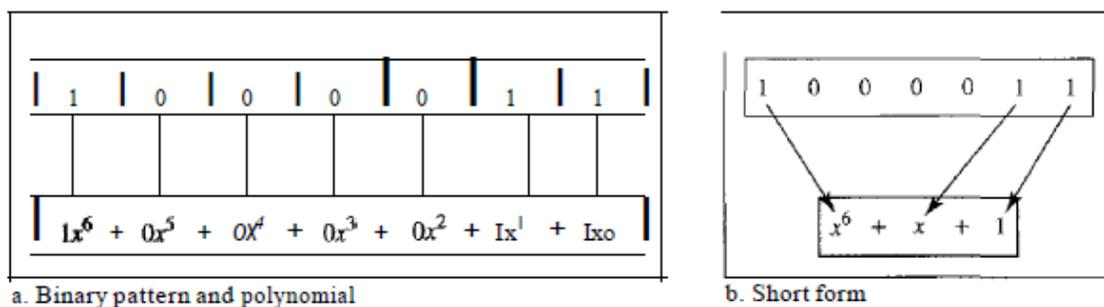
Figure 10.21 shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x23 + X3 + 1$. Here the bit pattern is 24 bits in length (three Is and twenty-one Os) while the polynomial is just three terms.

PARITY CHECK CODE

The most familiar error-detecting code is the simple parity-check code. In this code, a *k-bit* dataword is changed to an n-bit codeword where $n = k + 1$. The extra bit, called the parity bit, is selected to make the total number of Is in the codeword even. Although some implementations specify an odd number of 1's. The minimum Hamming distance for this category is dmin =2, which means that the code is a single-bit error-detecting code; it cannot correct any error.

Our first code (Table 10.1) is a parity-check code with k -= 4 and n =5.

Table 10.3 Simple parity-check code C(5, 4)

Datawords	Codewords	Datawords	Codewords
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver). The encoder uses a generator that takes a copy of a 4-bit dataword *(ao, aI' a2'* and *a3)* and generates a parity bit *roo* The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of Is in the codeword even.

Figure 10.10 Encoder and decoder for simple parity-check code
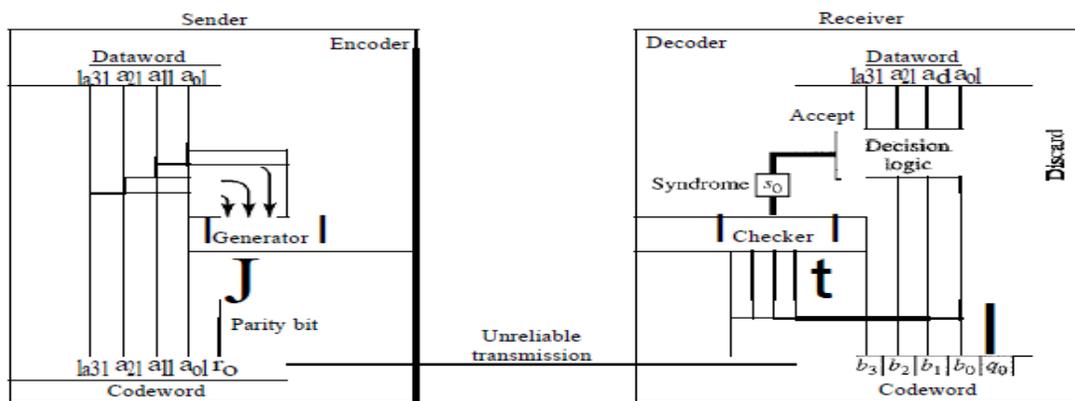
This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

$$r_0 = a_3 + a_2 + a_1 + a_0 \quad \text{(modulo-2)}$$

If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even. The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the syndrome, is just 1 bit. The syndrome is 0 when the number of Is in the received codeword is even; otherwise, it is 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \quad \text{(modulo-2)}$$

The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no error in the received codeword; the data portion of the received codeword is accepted as the dataword; if the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

Noiseless Channel and Noisy Channel Protocol:

Noiseless Channel

An ideal channel in which no frames are lost, duplicated or corrupted is regarded as Noiseless Channel.

Simplest Protocol

- In simplest protocol, there is no flow control and error control mechanism. It is a unidirectional protocol in which data frames travel in only one direction (from sender to receiver).
- Also, the receiver can immediately handle any received frame with a processing time that is small enough to be negligible.

- The protocol consists of two distinct procedures :a sender and receiver. The sender runs in the data link layer of the source machine and the receiver runs in the data link layer of the destination machine. No sequence number or acknowledgements are used here.

Stop and Wait Protocol

- The simplest retransmission protocol is stop-and-wait.
- Transmitter (Station A) sends a frame over the communication line and then waits for a positive or negative acknowledgement from the receiver (station B).
- If no error occurs in the transmission, station B sends a positive acknowledgement (ACK) to station A.
- Now, the transmitter starts to send the next frame. If frame is received at station B with errors, then a negative acknowledgement(NAK) is sent to station A. In this case, station 'A' must retransmit the old packet in a new frame.
- There is also a possibility that the information frames or ACKs may get lost.
- Then, the sender is equipped with a timer. If no recognizable acknowledgement is received when the timer expires at the end of time out interval, the same frame is sent again.
- The sender which sends one frame and then waits for an acknowledgement before process is known as **stop and wait.**
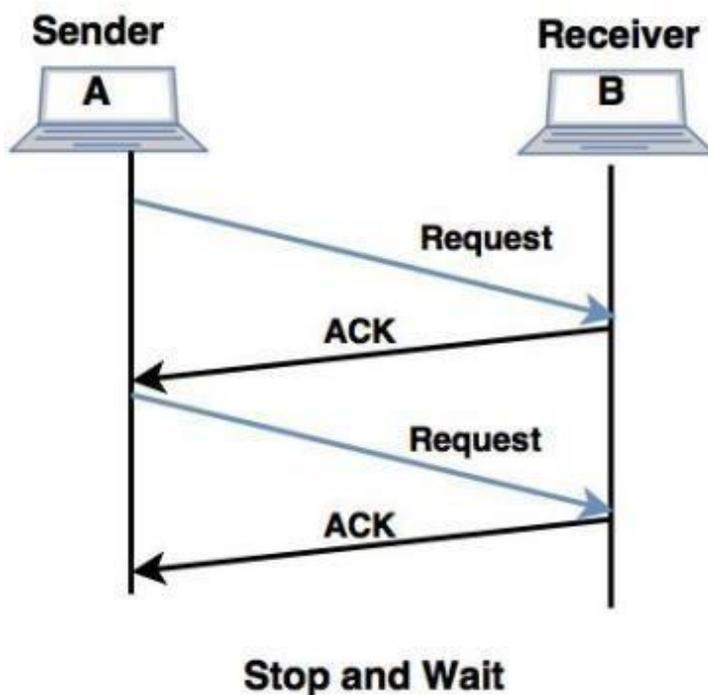
Stop and Wait

Noisy Channels

Consider the normal situation of a communication channel that makes errors. Frames may be either damaged or lost completely.

1. Stop and Wait Automatic Repeat Request

- In a noisy communication channel, if a frame is damaged in transit, the receiver hardware will detect this when it computes the checksum.

- If a damaged frame is received, it will be discarded and transmitter will retransmit the same frame after receiving a proper acknowledgement.

- If the acknowledgement frame gets lost and data link layer on 'A' eventually times out. Not having received an ACK, it assumes that its data frame was lost or damaged and sends the frame containing packet 1 again. This duplicate frame also arrives at data link layer on 'B', thus part of file will be duplicated and protocol is said to be failed.

- A typical approach to solve this problem is the provision of a sequence number in the header of the message.

- The receiver can then check the sequence number determine if the message is a duplicate since only message is transmitted at any time.

- The sending and receiving station needs only 1-bit alternating sequence of '0' or '1'to maintain the relationship of the transmitted message and its ACK/ NAK.

- A modulo-2 numbering scheme is used where the frames are alternatively label with '0' or '1' and positive acknowledgements are of the form ACK 0 and ACK 1.

2. Sequence numbers

- The protocol specifies that frames need to be numbered. This is done by using sequence number. A field is added to the data frame to hold the sequence number of that frame.

- The sequence numbers are based on modulo-2 arithmetic.

- Stop-and-wait ARQ is the simplest mechanism for error and flow control.

HDLC

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

Configurations and Transfer Modes

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

Normal Response Mode

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands; a secondary station can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in Figure 11.25.

Asynchronous Balanced Mode

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

11.26. This is the common mode today.

Figure 11.25 *Normal response mode*
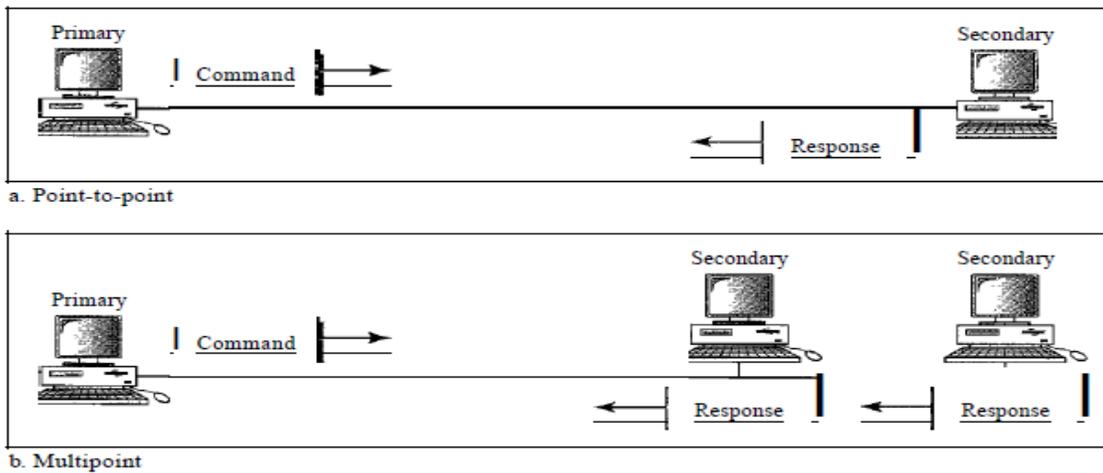
Figure 11.26 *Asynchronous balanced mode*
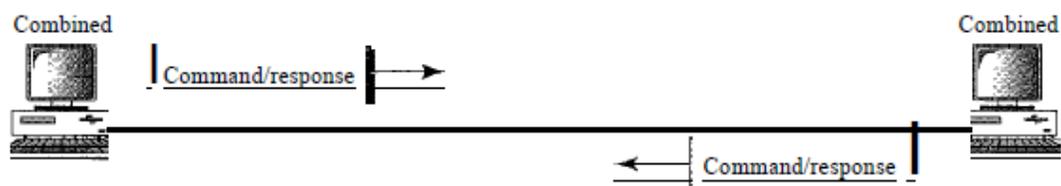
Frames

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

Frame Format

Each frame in HDLC may contain up to six fields, as shown in Figure 11.27: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field,and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

Figure 11.27 HDLC frames
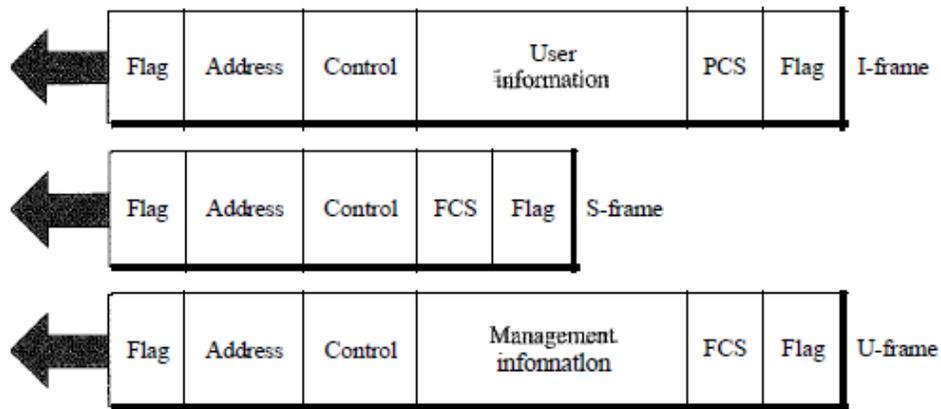
Fields

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

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

o **Address field**. The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary creates the frame, it contains *a from* address. An address field can be 1 byte or several bytes long, depending on the needs of the network. One byte can identify up to 128 stations (l bit is used for another purpose). Larger networks require multiple-byte address fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0; only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come.

o **Control field**. The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type.

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

o **FCS field**. The frame check sequence (FCS) is the HDLC error detection field.

Control Field

The control field determines the type of frame and defines its functionality. The format is specific for the type of frame, as shown in Figure 11.28.

Figure 11.28 *Control field format for the different frame types*
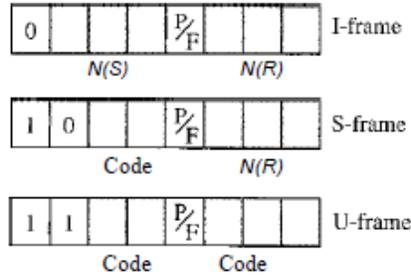
Control Field for I-Frames

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

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate. S-frames do not have information fields. If the first 2 bits of the control field is 10, this means the frame is an S-frame. The last 3 bits, called *N(R),* corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame. The 2 bits called code is used to define the type

of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

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

o **Receive not ready (RNR)**. If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of NCR is the acknowledgment number.

o **Reject (REJ).** If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in *Go-Back-N* ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of $NCR)$ is the negative acknowledgment number.

o **Selective reject (SREJ).** If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat.* The value of $N(R)$ is the negative acknowledgment number.

Control Field for V-Frames

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

Point-to-point protocol

A point-to-point connection provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other

options, such as microwave or satellite links, are also possible. When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

UNIT – III

Multiple Access

Random Access

ALOHA:

In the 1970s, Norman Abramson and his colleagues at the University of Hawaii devised a new and elegant method to solve the channel allocation problem. Their work has been extended by many researchers since then (Abramson, 1985).

Although Abramson's work, called the ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel. There are two versions of ALOHA: pure and slotted. They differ with respect to whether time is divided into discrete slots into which all frames must fit. Pure ALOHA does not require global time synchronization; slotted ALOHA does.

Pure ALOHA:

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. However, due to the feedback property of broadcasting, a sender can always find out whether its frame was destroyed by listening to the channel, the same way other users do. With a LAN, the feedback is immediate; with a satellite, there is a delay of 270 msec before the sender knows if the transmission was successful. If listening while transmitting is not possible for some reason, acknowledgements are needed. If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as contention systems.
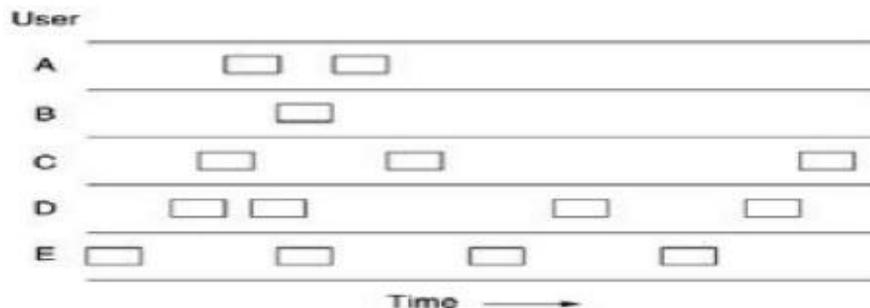
Fig.1 In pure ALOHA, frames are transmitted at completely arbitrary times.

A sketch of frame generation in an ALOHA system is given in Fig.1. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable length frames.

Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted later. The checksum cannot (and should not) distinguish between a total loss and a near miss.

Let the "frame time" denote the amount of time needed to transmit the standard, fixed length frame (i.e., the frame length divided by the bit rate). At this point we assume that the infinite population of users generates new frames according to a Poisson distribution with mean N frames per frame time. (The infinite-population assumption is needed to ensure that N does no decrease as users become blocked.) If $N > 1$, the user community is generating frames at a higher rate than the channel can handle, and nearly every frame will suffer a collision.

For reasonable throughput we would expect $0 < N < 1$. In addition to the new frames, the stations also generate retransmissions of frames that previously suffered collisions. Let us further assume that the probability of k transmission attempts per frame time, old and new combined, is also Poisson, with mean G per frame time. Clearly, $G \geq N$. At low load (i.e., N 0), there will be few collisions, hence few retransmissions, so G N. At high load there will be many collisions, so $G > N$. Under all loads, the throughput, S, is just the offered load, G, times the probability, P_0, of a transmission succeeding—that is, $S = GP_0$, where P_0 is the probability that a frame does not suffer a collision.

A frame will not suffer a collision if no other frames are sent within one frame time of its start, as shown in Fig.2.
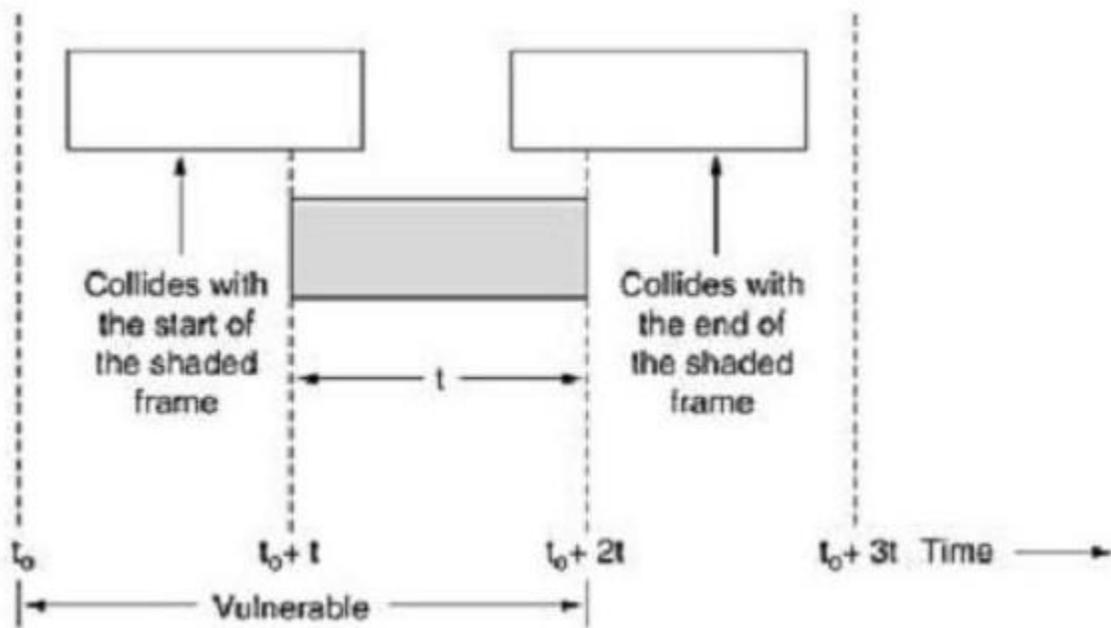
Fig.2. Vulnerable period for the shaded frame

Under what conditions will the shaded frame arrive undamaged? Let t be the time required to send a frame. If any other user has generated a frame between time t0 and t0+tr, the end of that frame will collide with the beginning of the shaded one. In fact, the shaded frame's fate was already sealed even before the first bit was sent, but since in pure ALOHA a station does not listen to the channel before transmitting, it has no way of knowing that another frame was already underway. Similarly, any other frame started between t0+t and t0+2t will bump into the end of the shaded frame.

The probability that k frames are generated during a given frame time is given by the Poisson distribution:

Equation

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

so the probability of zero frames is just e-G. In an interval two frame times long, the mean number of frames generated is 2G. The probability of no other traffic being initiated during the entire vulnerable period is thus given by P0 = e -2G. Using S = GP0, we get

$$S = Ge^{-2G}$$

The relation between the offered traffic and the throughput is shown in Fig. 4-3. The maximum throughput occurs at G = 0.5, with S = 1/2e, which is about 0.184. In other words, the best we can hope for is a channel utilization of 18 per cent. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100 per cent success rate.

Slotted ALOHA:

In 1972, Roberts published a method for doubling the capacity of an ALOHA system (Robert, 1972). His proposal was to divide time into discrete intervals, each interval corresponding to one frame. This approach requires the users to agree on slot boundaries. 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 return is 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 probability of no other traffic during the same slot as our test frame is e-G which leads to

\Equation

$$S = Ge^{-G}$$

As you can see from Fig.3, slotted ALOHA peaks at G = 1, with a throughput of S=1/e or about 0.368, twice that of pure ALOHA. If the system is operating at G = 1, the probability of an empty slot is 0.368. The best we can hope for using slotted ALOHA is 37 percent of the slots empty, 37 percent successes, and 26 percent collisions. Operating at higher values of G reduces the number of empties but increases the number of collisions exponentially.

To see how this rapid growth of collisions with G comes about, consider the transmission of a test frame. The probability that it will avoid a collision is e-G, the probability that all the other

users are silent in that slot. The probability of a collision is then just $1 – e^{-G}$. The probability of a transmission requiring exactly k attempts, (i.e., $k – 1$ collisions followed by one success) is
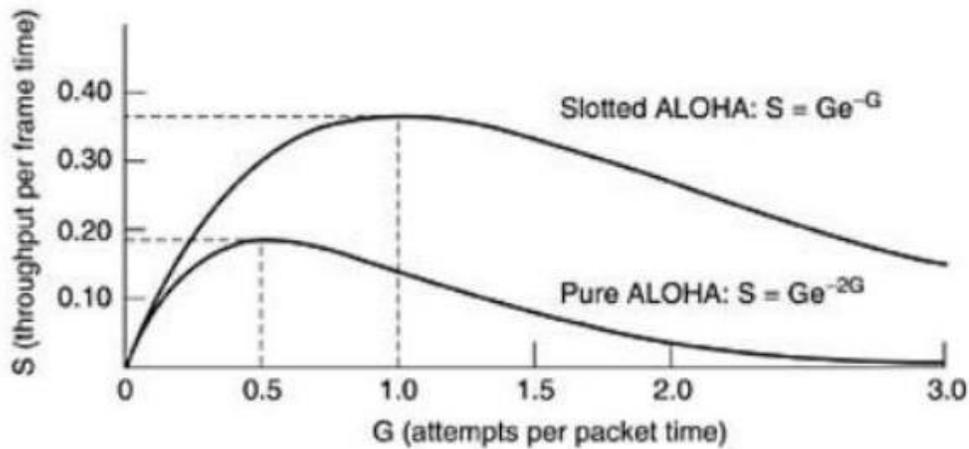
Fig.3 Throughput versus offered traffic for ALOHA systems.

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

The expected number of transmissions, E, per carriage return typed is then

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^G$$

As a result of the exponential dependence of E upon G, small increases in the channel load can drastically reduce its performance.

CDMA

Code-division multiple access (CDMA) was conceived several decades ago. Recent advances in electronic technology have finally made its implementation possible. CDMA differs from FDMA in that only one channel occupies the entire bandwidth of the link. It differs from TDMA in that all stations can send data simultaneously; there is no timesharing.

Let us first give an analogy. CDMA simply means communication with different codes. For example, in a large room with many people, two people can talk privately in English if nobody else understands English. Another two people can talk in Chinese if they are the only ones who understand Chinese, and so on. In other words, the common channel, the space of the room in this case, can easily allow communication between several couples, but in different languages (codes).

Idea

Let us assume we have four stations, 1, 2,3, and 4, connected to the' same channel. The data from station 1 are *d1*, from station 2 are *d2*, and so on. The code assigned to the first station is Cj, to the second is *c2*, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get O.

2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure.

Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get $d_,$. Cj. Station 2 multiplies its data by its code to get *d2* . <z- and so on. The data that go on the channel are the sum of all these terms, as shown in the box. Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiplies the data on the channel by C r- the code of station 1.

Because (cI . cI) is 4, but *(c2 . cl), (c3 . cl)'* and *(c4 . cI)* are all Os, station 2 divides the result by 4 to get the data from station 1.

Chips CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called *chips,* as shown in below figure. The codes are for the previous example.

Later in this chapter we show how we chose these sequences. For now, we need to know that we did not choose the sequences randomly; they were carefully selected. They are called *orthogonal sequences* and have the following properties:

Simple idea of communication with code

Data d_1

Station 1 $d_1 \cdot c_1$

Data d_2

Station 2 $d_2 \cdot c_2$

Data $d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4$

Common channel

$d_3 \cdot c_3$

Station 3 d_3

$d_4 \cdot c_4$

Station 4

d_4

Data

$$\text{data} = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1$$
$$= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1$$

Chip sequences

C_1	C_2	C_3	C_4
[+1 +1 +1 +1]	[+1 −1 +1 −1]	[+1 +1 −1 −1]	[+1 −1 −1 +1]
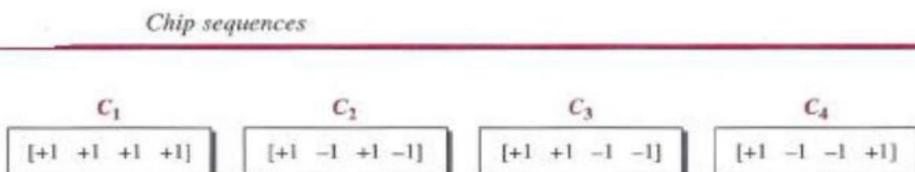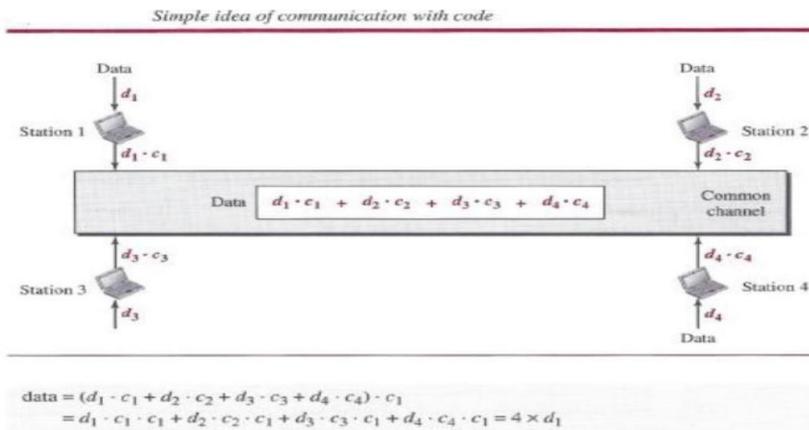
1. Each sequence is made of *N* elements, where *N* is the number of stations.

2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

2. [+1 +1 -1 -1] = [+2 +2 -2 -2]

3. If we multiply two equal sequences, element by element, and add the results, we get *N,* where *N* is the number of elements in each sequence. This is called the ***inner product*** of two equal sequences. For example, [+1 +1-1-1]. [+1 +1-1-1] = 1 + 1 + 1 + 1 =4

4. If we multiply two different sequences, element by element, and add the results, we get O. This is called the *inner product* of two different sequences. For example, [+ 1 +1 -1 -1] • [+1 +1 + 1 + 1] = 1 + 1 - 1 - 1 = 0

5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

[+1+1-1-1) + [+1+1+1+1)= [+2+2 0 0]

CSMA (Carrier Sense Multiple Access Protocols):

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

1.1-persistent CSMA:

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

The propagation delay has an important effect on the performance of the protocol. There is a small chance that just after a station begins sending, another station will become ready to send and sense the channel. If the first station's signal has not yet reached the second one, the latter will sense an idle channel and will also begin sending, resulting in a collision. The longer the propagation delay, the more important this effect becomes, and the worse the performance of the protocol. Even if the propagation delay is zero, there will still be collisions. If two stations become ready in the middle of a third station's transmission, both will wait politely until thetransmission ends and then both will begin transmitting exactly simultaneously, resulting in a collision. If they were not so impatient, there would be fewer collisions. Even so, this protocol is far better than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame. Intuitively, this approach will lead to a higher performance than pure ALOHA. Exactly the same holds for slotted ALOHA.

2. Non-persistent CSMA:

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

3. P-persistent CSMA:

The last protocol is **p-persistent CSMA**. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability q = 1 - p, it defers until the next slot. If that slot is also idle, It either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting. In the latter case, the unlucky station acts as if there had been a collision (i.e., it waits a random time and starts again). If the station initially senses the channel busy, it waits until the next slot and applies the above algorithm. Figure 4 shows the computed throughput versus offered traffic for all three protocols, as well as for pure and slotted ALOHA.
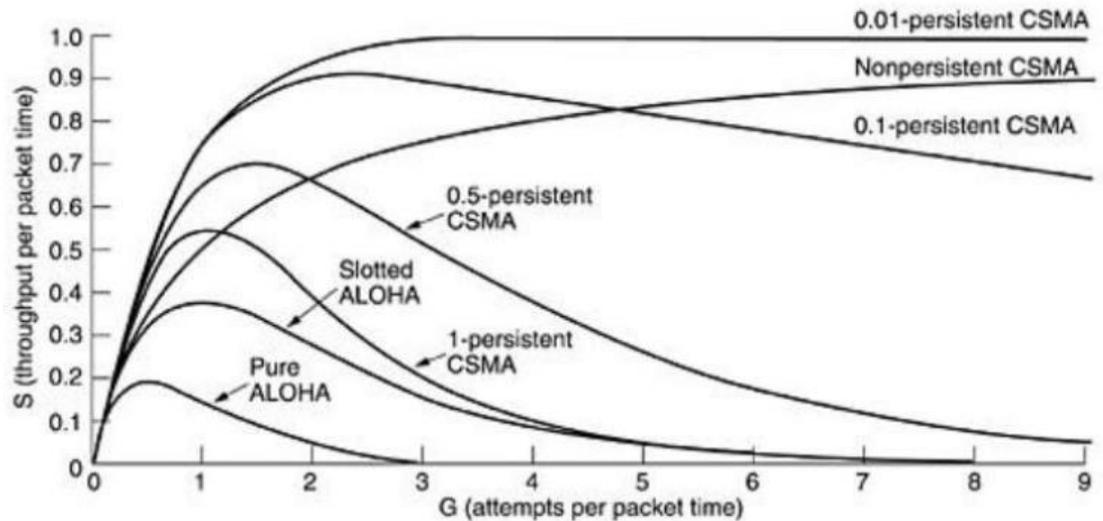
Fig.4 Comparison of the channel utilization versus load for various random access protocols

CSMA/CD (CSMA with Collision Detection)

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

This protocol, known as CSMA/CD (CSMA with Collision Detection) is widely used on LANs in the MAC sublayer. In particular, it is the basis of the popular Ethernet LAN, so it is worth devoting some time to looking at it in detail. CSMA/CD, as well as many other LAN protocols, uses the conceptual model of Fig.5. At the point marked t0, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision. Collisions can be detected by looking at the power or pulse width of the received signal and comparing it to the transmitted signal.
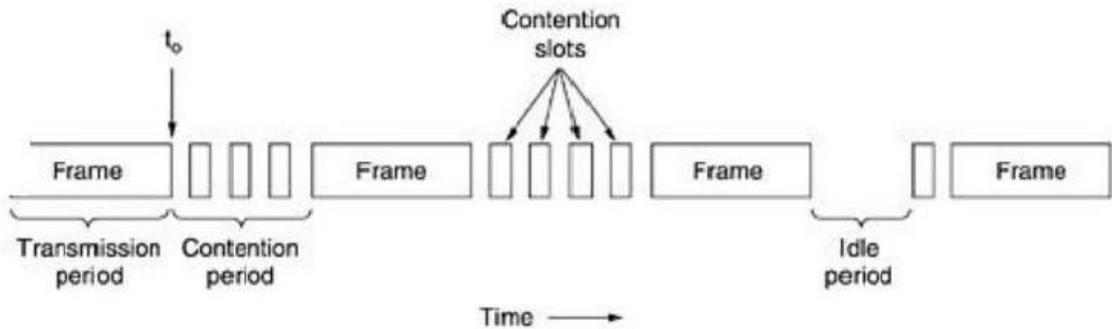
Fig.5. CSMA/CD can be in one of three states: contention, transmission, or idle

After a station detects a collision, it aborts its transmission, waits a random period of time, and then tries again, assuming that no other station has started transmitting in the meantime. Therefore, our model for CSMA/CD will consist of alternating contention and transmission periods, with idle periods occurring when all stations are quiet (e.g., for lack of work). Now let us look closely at the details of the contention algorithm. Suppose that two stations both begin transmitting at exactly time t0. How long will it take them to realize that there has been a collision? The answer to this question is vital to determining the length of the contention period and hence what the delay and throughput will be. The minimum time to detect the collision is then just the time it takes the signal to propagate from one station to the other.

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

CSMA/CA

Carrier sense **multiple** access **with collision avoidance** *(CSMA/CA)* was invented for wireless networks. Collisions are avoided through the use of *CSMA/CA's* three strategies: the inter frame space, the contention window, and acknowledgments, as shown in below figure.

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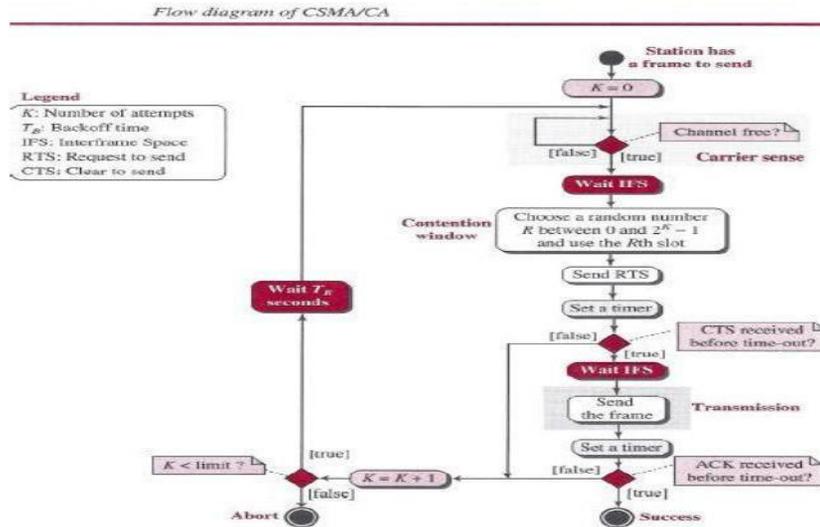
Inter frame Space (IFS) First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the *inter frame space* or *IFS*. Even though the channel may appear idle when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

Contention Window The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back off strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the p-persistent method except that a random outcome

defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time. See below figure.

Acknowledgment

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

Frame Exchange Time Line

Below figure shows the exchange of data and control frames in time.

1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.

a. The channel uses a persistence strategy with back off until the channel is idle.

b. After the station is found to be idle, the station waits for a period of time called the *DCF inter frame space (DIFS);* then the station sends a control frame called the *request to send (RTS).*

2. After receiving the RTS and waiting a period of time called the *short inter frame space (SIFS),* the destination station sends a control frame, called the *clear to send (CTS),* to the source station. This control frame indicates that the destination station is ready to receive data.
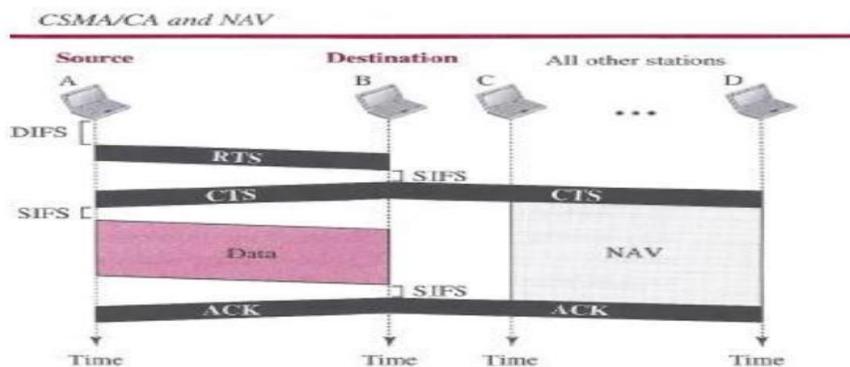
3. The source station sends data after waiting an amount of time equal to SIFS.

4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in *CSMA/CD* is a kind of indication to the source that data have arrived.

Controlled Access

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three controlled-access methods.

Reservation

In the reservation method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval. If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame. Below figure shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.
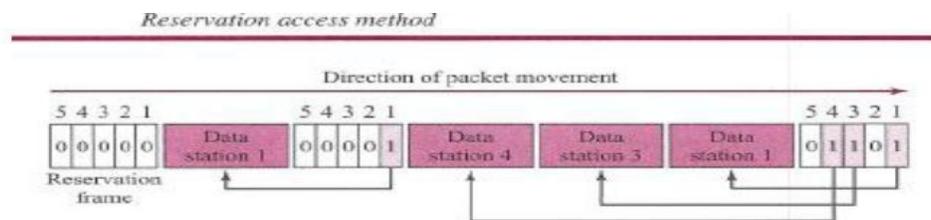
Polling

Polling works with topologies in which one device is designated as a *primary station* and the other devices are *secondary stations*. All data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session This method uses poll and select functions to prevent collisions.

However, the drawback is if the primary station fails, the system goes down.

Select and poll functions in polling-access method
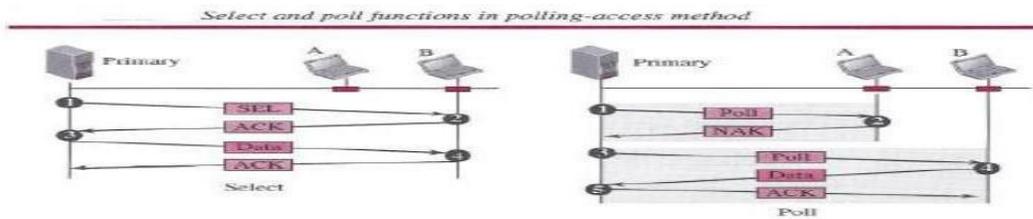
Select

       The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available. If it has something to send, the primary device sends it. What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll

       The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

Token Passing

       In the token-passing method, the stations in a network are organized in a logical ring.In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right

to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

But how is the right to access the channel passed from one station to another? In this method, a special packet called a *token* circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station.

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed. For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Below figure shows four different physical topologies that can create a logical ring.

Logical ring and physical topology in token-passing access method

a. Physical ring   b. Dual ring   c. Bus ring   d. Star ring
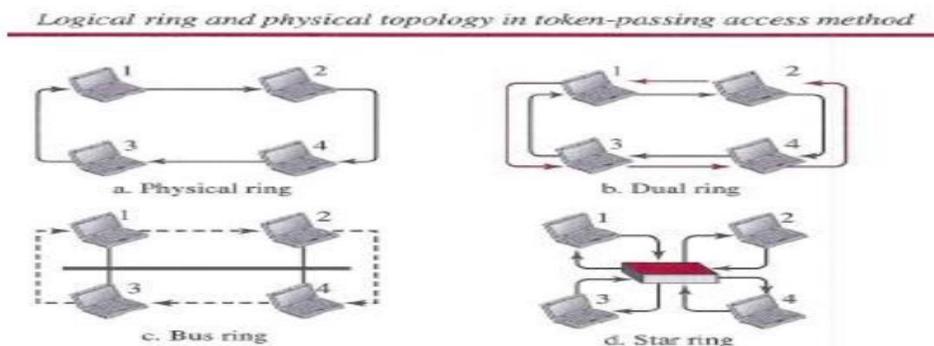
In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not

have to have the address of the next successor. The problem with this topology is that if one of the links-the medium between two adjacent stations-fails, the whole system fails. The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called *FDDI (Fiber Distributed Data Interface]* and *CDDI (Copper Distributed Data Interface]* use this topology. In the bus ring topology, also called a token bus, the stations are connected to a single cable called a *bus.* They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology. In a star and ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

Channelization

Channelization (or *channel partition,* as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

FDMA

In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a band pass filter to confine the transmitter frequencies. To prevent station interferences, the

allocated bands are separated from one another by small *guard bands. Below f*igure shows the idea of FDMA.

Frequency-division multiple access (FDMA)
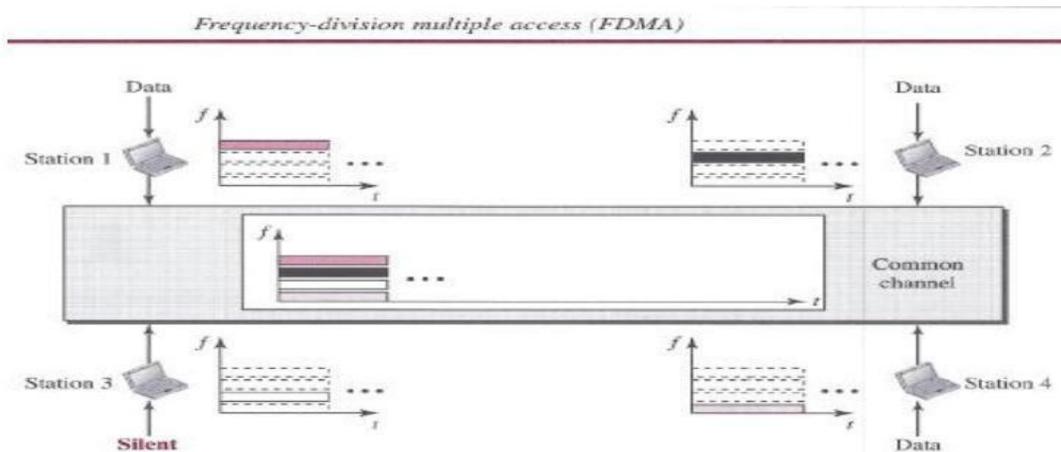
FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA. We will see in Chapter 16 how this feature can be used in cellular telephone systems. We need to emphasize that although FDMA and frequency-division multiplexing (FDM) conceptually seem similar, there are differences between them FDM. The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a band pass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data-link layer. The data link layer in each station tells its physical layer to make a band pass signal from the data passed to it. The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically band pass-filtered. They are mixed when they are sent to the common channel.

TDMA

Time-division multiple access (TDMA)
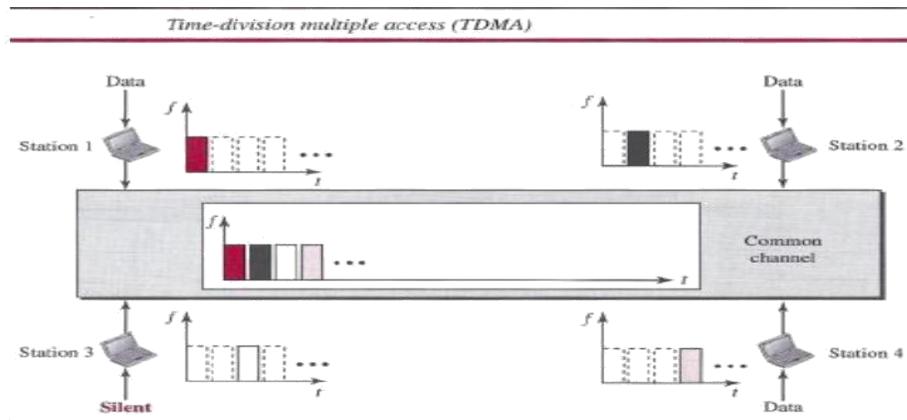
In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. Below figure shows the idea behind TDMA.

The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area. To compensate for the delays, we can insert *guard times.* Synchronization is normally accomplished by having some synchronization bits (normally referred to as *preamble bits)* at the beginning of each slot. We also need to emphasize that although TDMA and time-division multiplexing (TDM) conceptually seem the same, there are differences between them. TDM, as, is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel. TDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

Wired LANs: IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers.

Project 802 does not seek to replace any part of the OSI or the Internet model. Instead, it is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

The standard was adopted by the American National Standards Institute (ANSI). In 1987, the International Organization for Standardization (ISO) also approved it as an international standard under the designation ISO 8802. The relationship of the 802 Standard to the traditional OSI model is shown in Figure 1.

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

logical link control (LLC) and media access control (MAC).

IEEE has also created several physical layer standards for different LAN protocols.
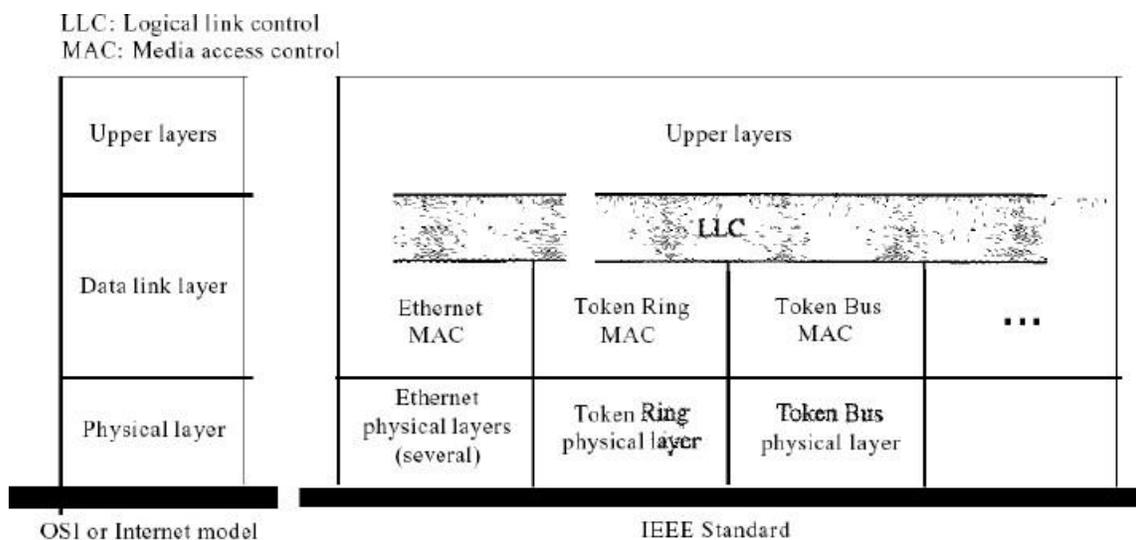
Figure 1 *IEEE standard for LANs*

Data Link Layer

As we mentioned before, the data link layer in the IEEE standard is divided into two sublayers: LLC and MAC. *Logical Link Control (LLC)*

We said that data link control handles framing, flow control, and error control. In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control. Framing is handled in both the LLC sublayer and the MAC sublayer.

The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LANs. A single LLC protocol can provide interconnectivity between different LANs

because it makes the MAC sublayer transparent. Figure 1 shows one single LLC protocol serving several MAC protocols. Framing LLC defines a protocol data unit (PDU) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the upper-layer protocol at the source and destination that uses LLC. These fields are called the destination service access point (DSAP) and the source service access point (SSAP). The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. In other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sublayer, as shown in Figure 2.

Need for LLC The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services. For example, if a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols. However, most upper-layer protocols such as IP, do not use the services of LLC.
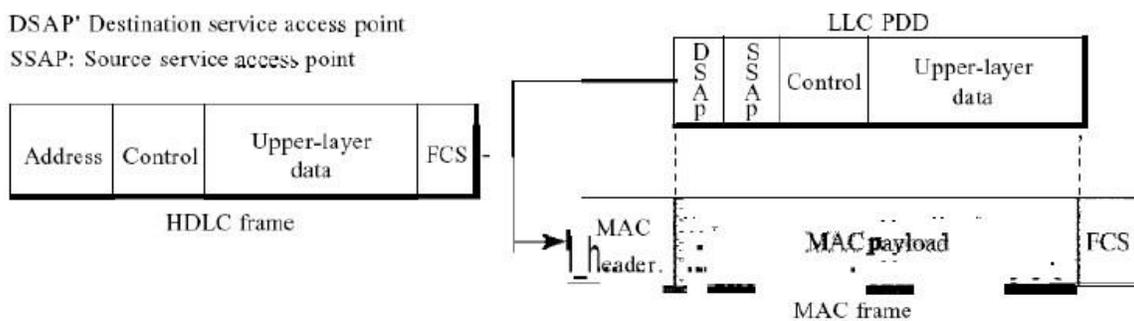
Figure 13.2 *HDLCframe compared with LLC and MAC frames*

Media Access Control (MAC)

IEEE Project 802 has created a sublayer called media access control that defines the specific access method for each LAN. For example, it defines *CSMA/CD* as the media access method for Ethernet LANs and the tokenpassing method for Token Ring and Token Bus LANs. As we discussed in the previous section, part of the framing function is also handled by the MAC layer. In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.

Physical Layer

The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specifications for each Ethernet implementations.

STANDARD ETHERNET
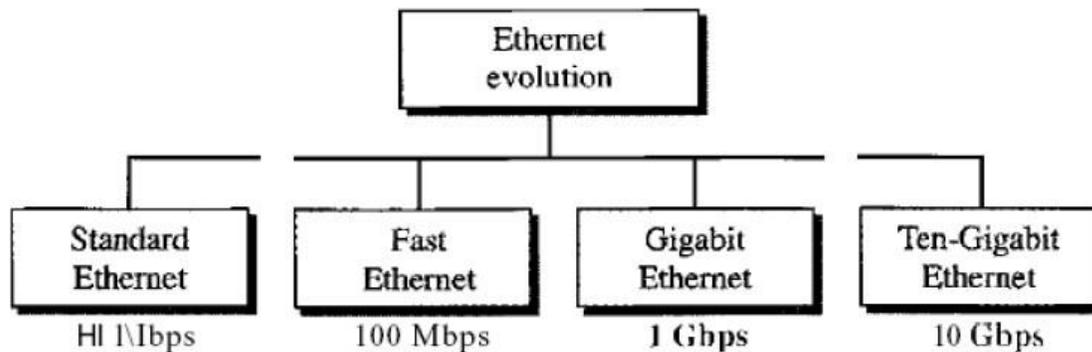
Figure 3 *Ethernet evolution through four generations*

MAC Sublayer

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

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRe. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers. The format of the MAC frame is shown in Figure 4.

Preamble: 56 bits of alternating 1s and as.

SFD: Start frame delimiter, flag (10101011)

Preamble	SFD	Destination address	Source address	Length or type	Data and padding	CRC
7 bytes	1 byte. 1	6 bytes	6 bytes	2 bytes		4 bytes

Physical layer
header
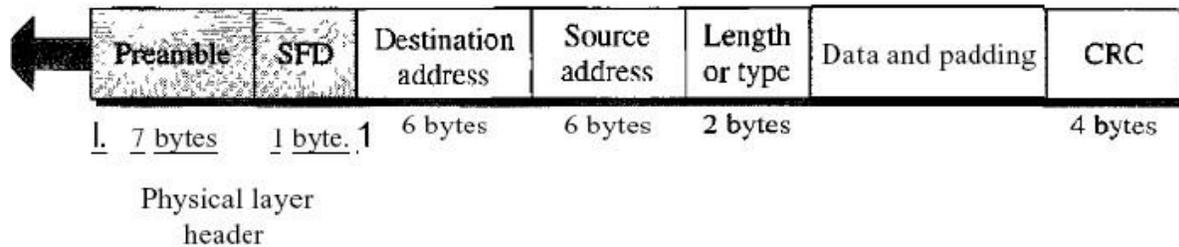
Figure 4 *802.3 MACframe*

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

D Start frame delimiter (SFD). The second field (l byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.

Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet. We will discuss addressing shortly.

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

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

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

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame,

as shown in Figure 5.

| Minimum payload length: 46 bytes |
| Maximum payload length: 1500 bytes |

Destination address	Source address	Length PDU	Data and padding	CRC
6 bytes	6 bytes	2 bytes		4 bytes
Minimum frame length: 512 bits or 64 bytes				

MaXImum frame length. 12,144 bIts or 1518 bytes

Figure 5 *Minimum and maximum lengths*

The minimum length restriction is required for the correct operation of *CSMAlCD*. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 - 18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference.

The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Frame length:

Minimum: 64 bytes (512 bits) Maximum: 1518 bytes (12,144 bits)

Addressing

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

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

6 bytes =12 hex digits =48 bits

Figure 6 *Example ofan Ethernet address in hexadecimal notation*

Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address-the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. Figure 7 shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.
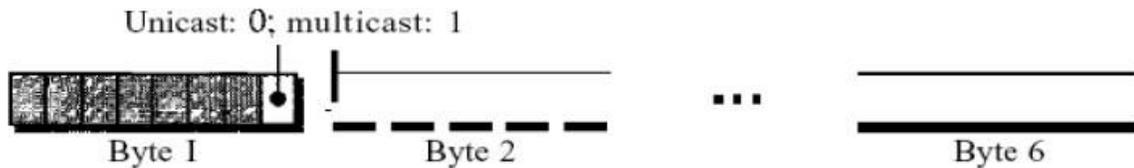
Figure 7 *Unicast and multicast addresses*

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

6.2.2 Physical Layer

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

Encoding and Decoding

All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the received signal is interpreted as Manchester and decoded into data. Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure 9 shows the encoding scheme for Standard Ethernet.
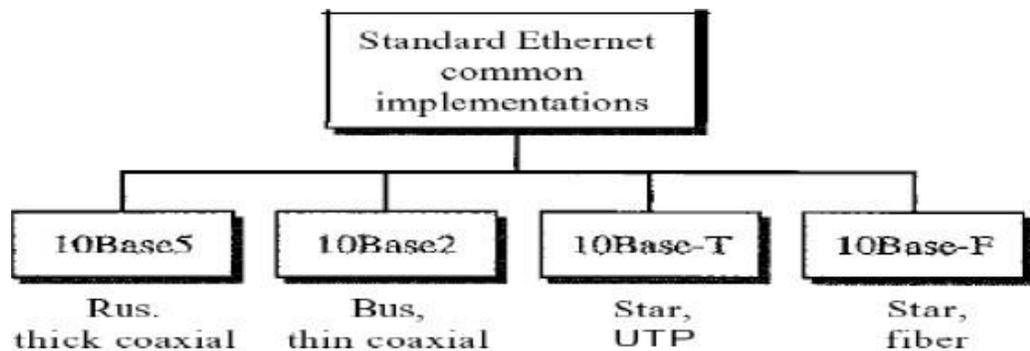
Figure 8 *Categories ofStandard Ethernet*

Fast Ethernet

Fast Ethernet was designed to compete with LAN protocols such as FDDI or Fiber Channel (or Fibre Channel, as it is sometimes spelled). IEEE created Fast Ethernet under the name 802.3u. Fast Ethernet is backward-compatible with Standard Ethernet, but it can transmit data 10 times faster at a rate of 100 Mbps.

The goals of Fast Ethernet can besummarized as follows:

1. Upgrade the data rate to 100 Mbps.

2. Make it compatible with Standard Ethernet.

3. Keep the same 48-bit address.

4. Keep the same frame format.

5. Keep the same minimum and maximum frame lengths.

Gigabit Ethernet

The technology is based on fiber optic cable. Multi-mode fiber is able to transmit a gigabit rate to at least 580 meters and with single-mode runs exceeding 3km.

Wireless LAN IEEE 802.11

Architecture

The standard defines two kinds of services: the basic service set (BSS) and the extended service set (ESS).

Basic Service Set

IEEE 802.11 defines the basic service set (BSS) as the building block of a wireless LAN. A basic service set is made of stationary or mobile wireless stations and an optional central base station, known as the access point (AP). Figure 9 shows two sets in this standard.

The BSS without anAP is a stand-alone network and cannot send data to other BSSs. It is called an *ad hoc architecture.* In this architecture, stations can form a network without the need of an AP; they can locate one another and agree to be part of a BSS. A BSS with an AP is sometimes referred to as an *infrastructure* network.
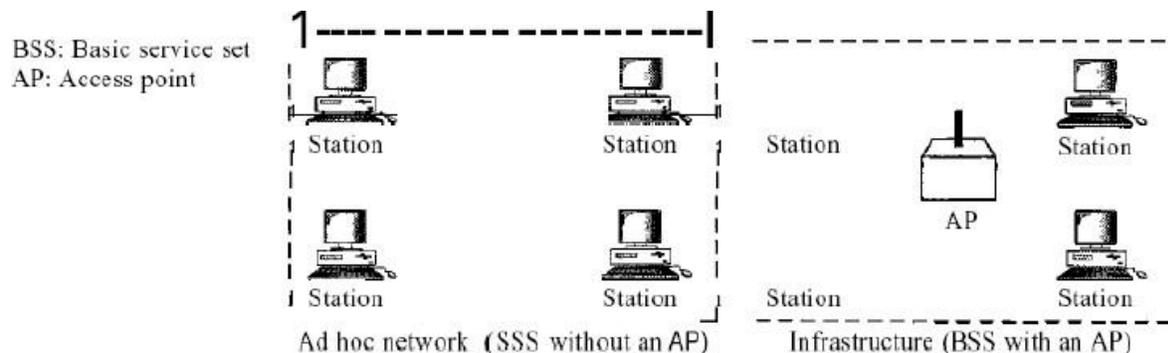
Figure. 9 *Basic service sets (BSSs)*

Extended Service Set

An extended service set (ESS) is made up of two or more BSSs with APs. In this case, the BSSs are connected through a *distribution system,* which is usually a wired LAN. The distribution system connects the APs in the BSSs. IEEE 802.11 does not restrict the distribution system; it can be any IEEE LAN such as an Ethernet. Note that the extended service set uses two types of stations: mobile and stationary. The mobile stations are normal stations inside a BSS. The stationary stations are AP stations that are part of a wired LAN. Figure 10 shows an ESS.

ESS: Extended service set
BSS: Basic service set
AP: Access point

Distribution system

Server or Gateway

BSS   BSS   BSS
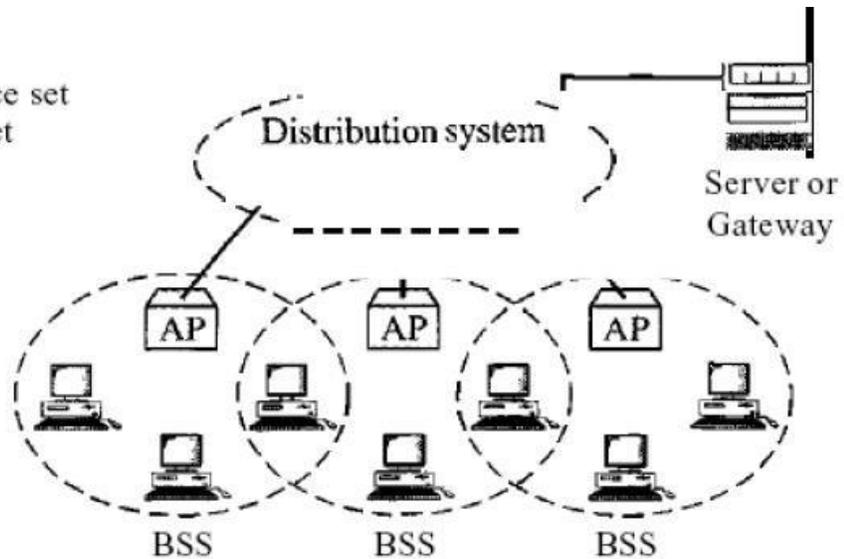
Figure 10 *Extended service sets (ESSs)*

Bluetooth

Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers (desktop and laptop), cameras, printers, coffee makers, and so on. A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously; the devices, sometimes called gadgets, find each other and make a network called a piconet. A Bluetooth LAN can even be connected to the Internet if one of the gadgets has this capability. A Bluetooth LAN, by nature, cannot be large. If there are many gadgets that try to connect, there is chaos.

Bluetooth technology has several applications. Peripheral devices such as a wireless mouse or keyboard can communicate with the computer through this technology. Monitoring devices can communicate with sensor devices in a small health care center. Home security devices can use this technology to connect different sensors to the main security controller. Conference attendees can synchronize their laptop computers at a conference.

Bluetooth was originally started as a project by the Ericsson Company. It is named for Harald Blaatand, the king of Denmark (940-981) who united Denmark and Norway. *Blaatand* translates to *Bluetooth* in English.

Today, Bluetooth technology is the implementation of a protocol defined by the IEEE 802.15 standard. The standard defines a wireless personal-area network (PAN) operable in an area the size of a room or a hall.

Architecture

Bluetooth defines two types of networks: piconet and scatternet.

Piconets

A Bluetooth network is called a piconet, or a small net. A piconet can have up to eight stations, one of which is called the primary;t the rest are called secondaries. All the secondary stations synchronize their clocks and hopping sequence with the primary. Note that a piconet can have only one primary station. The communication between the primary and the secondary can be one-to-one or one-to-many. Figure 1 shows a piconet.
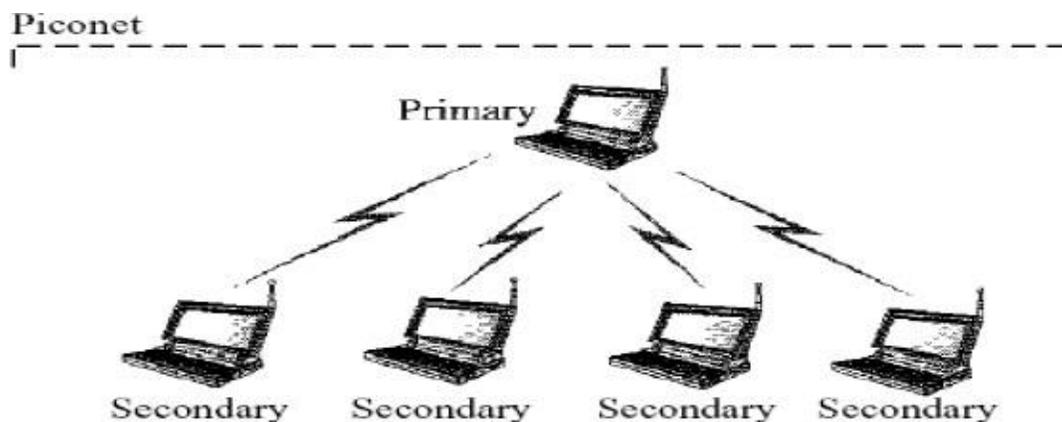
Figure 1 *Piconet*

Although a piconet can have a maximum of seven secondaries, an additional eight secondaries can be in the *parked state.* A secondary in a parked state is synchronized with the primary, but cannot take part in communication until it is moved from the parked state. Because only eight stations can be active in a piconet, activating a station from the parked state means that an active station must go to the parked state.

Scat/ernet

Piconets can be combined to form what is called a scatternet. A secondary station in one piconet can be the primary in another piconet. This station can receive messages from the primary in the first piconet (as a secondary) and, acting as a primary, deliver them to secondaries in the second piconet. A station can be a member of two piconets.

Piconet
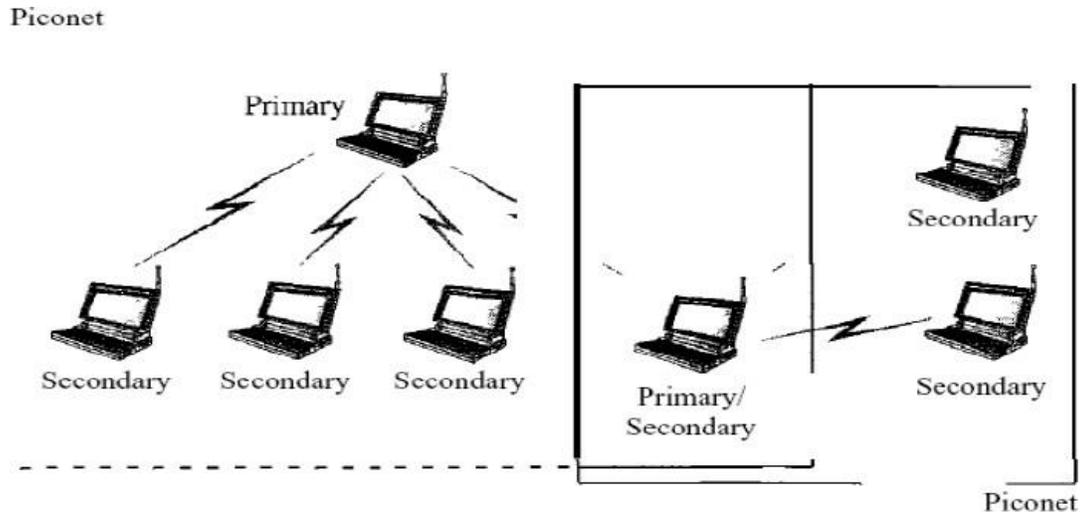
Figure 2 illustrates a scatternet.

Bluetooth Devices

A Bluetooth device has a built-in short-range radio transmitter. The current data rate is 1 Mbps with a 2.4-GHz bandwidth. This means that there is a possibility of interference between the IEEE 802.11b wireless LANs and Bluetooth LANs.

IEEE 802.16

The 802.16 standard essentially standardizes two aspects of the air interface – the physical layer (PHY) and the media access control (MAC) layer. This section provides an overview of the technology employed in these two layers in the mobile 802.16e specification.

PHY

802.16e uses scalable OFDMA to carry data, supporting channel bandwidths of between 1.25 MHz and 20 MHz, with up to 2048 subcarriers. It supports adaptive modulation and coding, so that in conditions of good signal, a highly efficient 64 QAM coding scheme is used, whereas when the signal is poorer, a more robust BPSK coding mechanism is used. In intermediate conditions, 16 QAM and QPSK can also be employed. Other PHY features include support for multiple-input multiple-output (MIMO) antennas in order to provide good non-line-of-sight propagation (NLOS) characteristics (or higher bandwidth) and hybrid automatic repeat request (HARQ) for good error correction performance.

Although the standards allow operation in any band from 2 to 66 GHz, mobile operation is best in the lower bands which are also the most crowded, and therefore most expensive

MAC

The 802.16 MAC describes a number of *Convergence Sublayers* which describe how wireline technologies such as Ethernet, Asynchronous Transfer Mode (ATM) and Internet Protocol (IP) are encapsulated on the air interface, and how data is classified, etc. It also describes how secure communications are delivered, by using secure key exchange during authentication, and encryption using Advanced Encryption Standard (AES) or Data Encryption Standard (DES) during data transfer. Further features of the MAC layer include power saving mechanisms (using *sleep mode* and *idle mode*) and handover mechanisms.

A key feature of 802.16 is that it is a connection-oriented technology. The subscriber station (SS) cannot transmit data until it has been allocated a channel by the base station (BS). This allows 802.16e to provide strong support for quality of service (QoS).

UNIT IV

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).
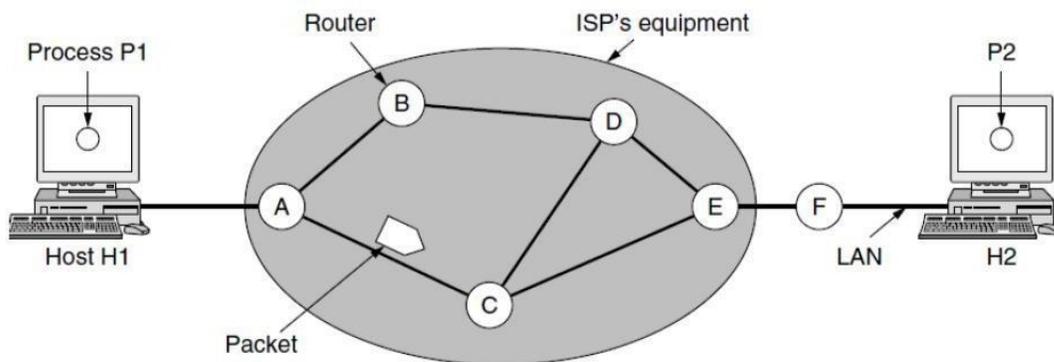
The environment of the network layer protocols.

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 destination host, 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: The services should be independent of the router technology.

The transport layer should be shielded from the number, type, and topology of the routers present.

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 the telephone system, and the network is called a **virtual-circuit network**. In this section, we will examine datagram networks; in the next one, we will examine virtual-circuit networks.

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 table is 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

Routing within a datagram network.
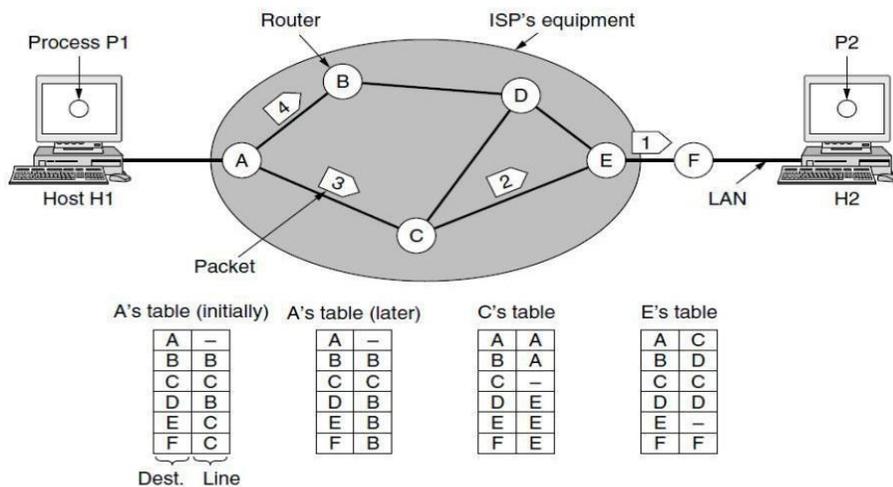
then to *F*. When it gets to *F*, it is sent within a frame over the LAN to *H2*. Packets 2 and 3 follow the same route. However, something different happens to packet 4. When it gets to *A* it is sent to

router *B*, even though it is also destined for *F*. For some reason, *A* decided to send packet 4 via a 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.
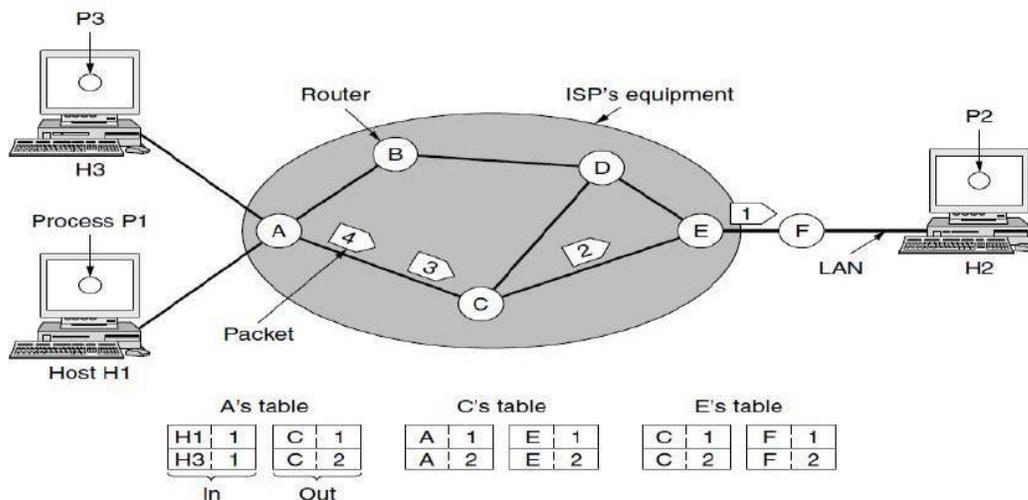
Routing within a virtual-circuit network.

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* assigns a 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.

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 amount of 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 datagrams 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 datagrams are used. Datagrams also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

PACKET SWITCHING

From the discussion of routing and forwarding in the previous section, we infer that a kind of *switching* occurs at the network layer. A router, in fact, is a switch that creates a connection between an input port and an output port (or a set of output ports), just as an electrical switch connects the input to the output to let electricity flow. Although in data communication switching techniques are divided into two broad categories, circuit switching and packet switching, only packet switching is used at the network layer because the unit of data at this layer is a packet. Circuit switching is mostly used lat the physical layer; the electrical switch mentioned earlier is a kind of circuit switch. We discussed circuit switching in Chapter 8; we discuss packet switching. At the Network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network. The source of the message sends the packets one by one; the destination of the message receives the packets one by one. The destination waits for all packets belonging to the same message to arrive before delivering the message to the upper layer. The connecting devices in a packet-switched network still need to decide how to route the packets to the final destination. Today, a packet-switched network can use two different approaches to route the packets: the *datagram approach* and the *virtual circuit approach.* We discuss both approaches in the next section.

Datagram Approach: Connectionless Service

When the Internet started, to make it simple, the network layer was designed to provide a connectionless service in which the network-layer protocol treats each packet independently, with each packet having no relationship to any other packet. The idea was that the network layer is only responsible for delivery of packets from the source to the destination. Id this approach, the packets in a message mayor may not travel the same path to their destination. Figure shows the idea. When the network layer provides a connectionless service, each packet traveling in the Internet is an independent entity; there is no relationship between packets belonging to the same message. The switches in this type of network are called *routers.* A packet belonging to a message may be followed by a packet belonging to the same message or to a different message. A packet may be followed by a packet coming from the same or from a different source.
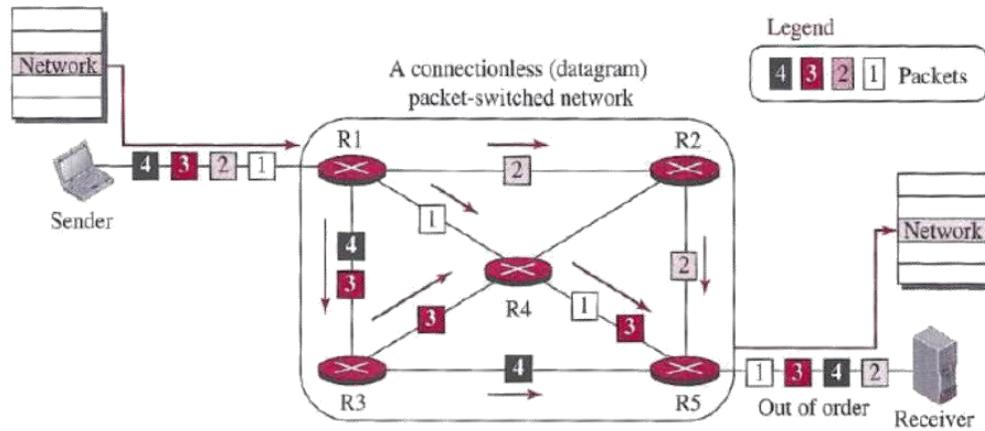
A connectionless packet-switched network

Each packet is routed based on the information contained in its header: source and destination addresses. The destination address defines where it should go; the source address defines where it comes from. The router in this case routes the packet based only on the destination address. The source address may be used to send an error message to the source if the packet is discarded. Figure 18.4 shows the forwarding process in a router in this case. We have used symbolic addresses such as A and B.
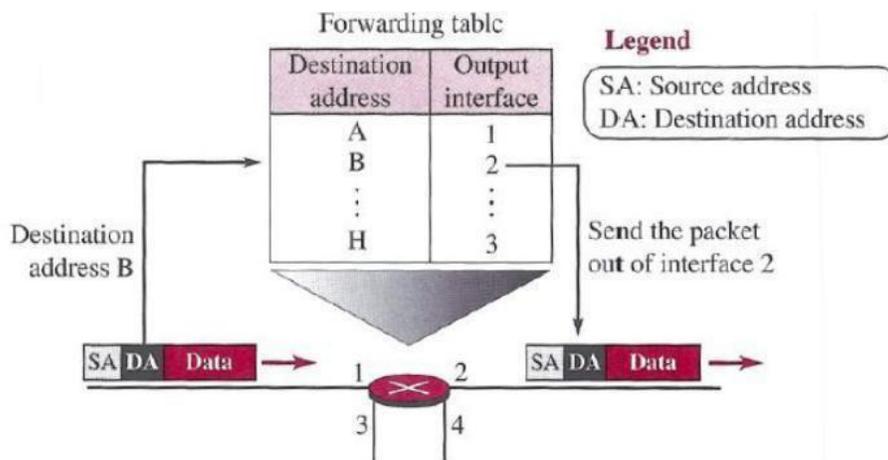
Forwarding process in a muter when used in a connection less network

In the datagram approach, the forwarding decision is based on the destination address of the packet.

Virtual-Circuit Approach: Connection-Oriented Service

In a connection-oriented service (also called *virtual-circuit approach),* there is a relationship between all packets belonging to a message. Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams can all follow the same path. In this type of service, not only must the packet contain the source and destination addresses, it must also contain a flow label, a virtual circuit identifier that defines the virtual path the packet should follow. Shortly, we will show how this flow label is determined, but for the moment, we assume that the packet carries this label. Although it looks as though the use of the label may make the source and destination addresses unnecessary during the data transfer phase, parts of the Internet at the network layer still keep these addresses. One reason is that part of the packet path may still be using the connectionless service. Another reason is that the protocol at the network layer is designed with these addresses, and it may take a while before they can be changed. Figure shows the concept of connection-oriented service.
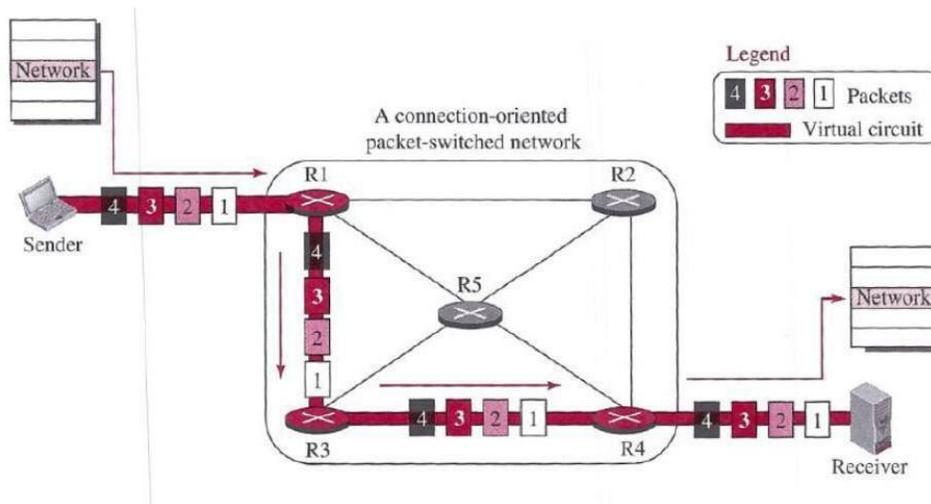
A virtual-circuit packet-switched network

Each packet is forwarded based on the label in the packet. To follow the idea of connection-oriented design to be used in the Internet, we assume that the packet has a label when it reaches

the router. Figure 18.6 shows the idea. In this case, the forwarding decision is based on the value of the label, or *virtual circuit identifier,* as it is sometimes called. To create a connection-oriented service, a three-phase process is used: setup, data transfer, and teardown. In the setup phase, the source and destination addresses of the sender and receiver are used to make table entries for the connection-oriented service. In the teardown phase, the source and destination inform the router to delete the corresponding entries. Data transfer occurs between these two phases.

Setup Phase

In the setup phase, a router creates an entry for a virtual circuit. For example, suppose source .A needs to create a virtual circuit to destination B. Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.
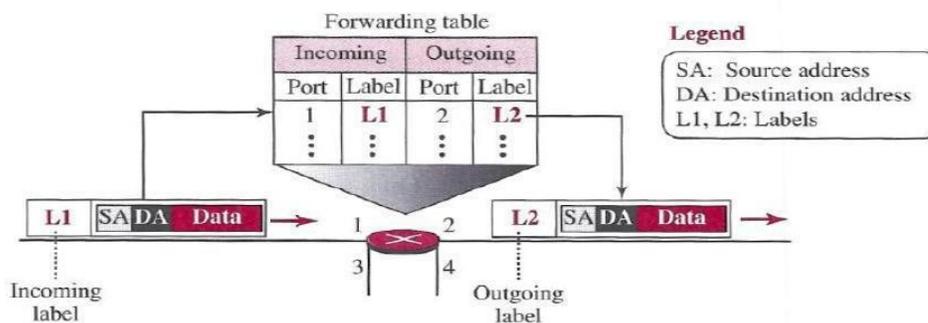
Forwarding process in a router when used in a virtual-circuit

network ***Request packet***

A request packet is sent from the source to the destination. This auxiliary packet carries the source and destination addresses. Figure shows the process.

A to B			
Incoming		Outgoing	
Port	Label	Port	Label
1	14	3	

Legend

A to B   Request packet

          Virtual circuit

Incoming		Outgoing	
Port	Label	Port	Label
1	66	3	

A to B

Incoming		Outgoing	
Port	Label	Port	Label
1	22	4	
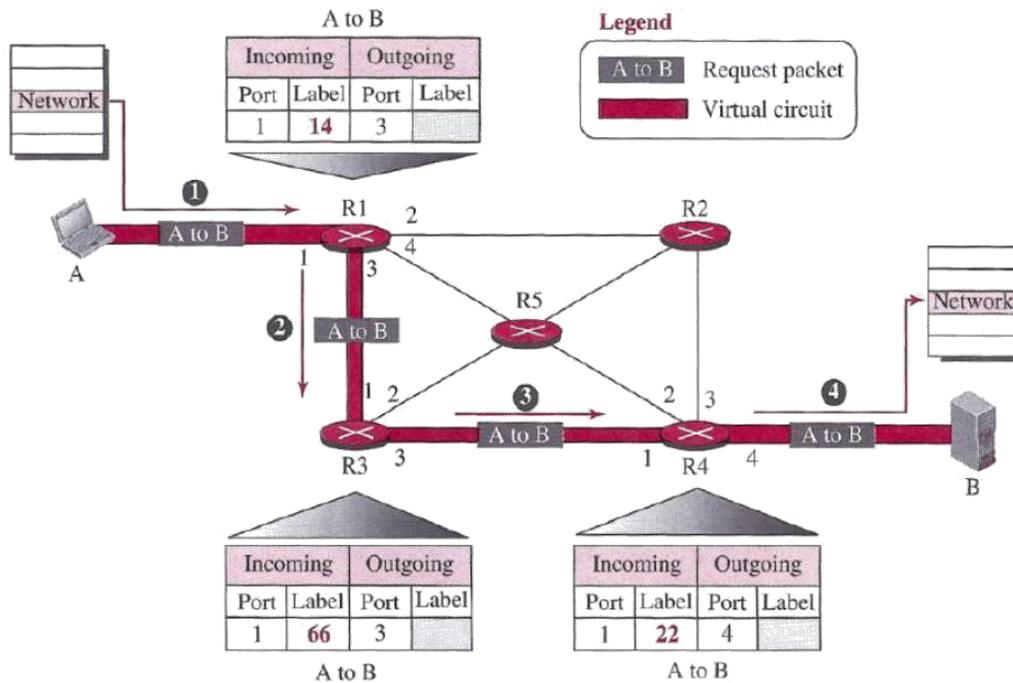
A to B

Sending request packet in a virtual-circuit network

Source A sends a request packet to router Rl.

Router Rl receives the request packet. It knows that a packet going from A to B goes 0rt through port 3. How the router has obtained this information is a point covered later. For the moment, assume that it knows the output port. The router creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The router assigns the incoming port (1) and chooses an available in rooming label (14) and the outgoing port (3). It does not yet know the outgoing label, which will be found during the acknowledgment step. The router then forwards the packet through port 3 to router R3.

1 Router iR3 receives the setup request packet. The same events happen here as at router Rl; three columns of the table are completed: in this case, incoming port (1)incoming label (66), and outgoing port (3).

2 Router R4 receives the setup request packet. Again, three columns are completed: incoming port (1), incoming label (22), and outgoing port (4).

3 Destination B receives the setup packet, and if it is ready to receive packets from A, it assigns a label to the incoming packets that come from A, in this case 77, as shown in Figure. This label lets the destination know that the packets come from A,I and not from other sources.

Acknowledgment Packet

A special Packet, called the acknowledgment packet, completes the entries in the switching tables. Figure shows the process.

Sending acknowledgments in a virtual-circuit network

1 The destination sends an acknowledgment to router R4. The acknowledgment carries the global source and destination addresses so the router knows which entry in the table is to be completed. The packet also carries label 77, chosen by the destination as the incoming label for packets from A. Router R4 uses this label to complete the outgoing label column for this entry. Note that 77 is the incoming label for destination B, but the outgoing label for router R4.
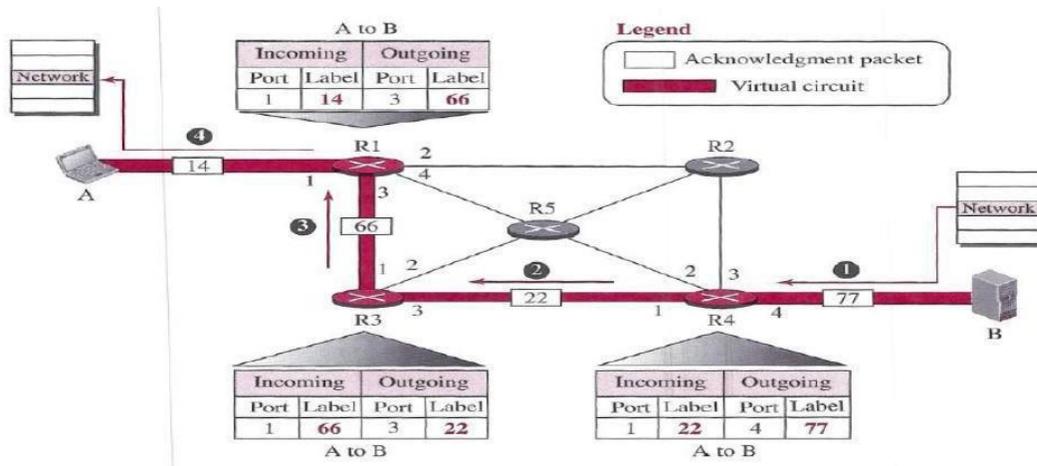
2 Router R4 sends an acknowledgment to router R3 that contains its incoming label in the table, chosen in the setup phase. Router R3 uses this as the outgoing label in the table.

3 Router R3 sends an acknowledgment to router Rl that contains its incoming label in the table, chosen in the setup phase. Router RI uses this as the outgoing label in the table.

4 Finally router Rl sends an acknowledgment to source A that contains its incoming label in the table, chosen in the setup phase.

5 The source uses this as the outgoing label for the data packets to be sent to destination B.

Data- Transfer Phase

The second phase is called the data-transfer phase. After all routers have created their forwarding table for a specific virtual circuit, then the network-layer packets belonging to one message can be sent one after another. In Figure, we show the flow of a single packet, but the process is the same for 1, 2, or 100 packets. The source computer uses the label 14, which it has received from router Rl in the setup.
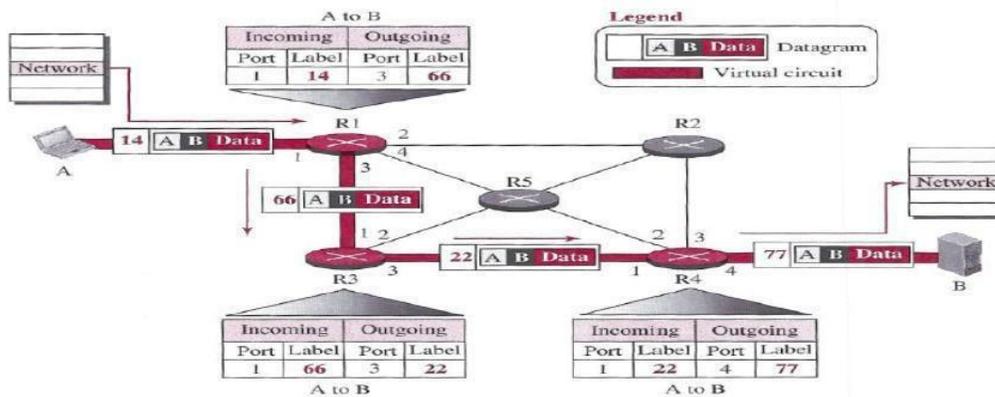
Flow of one packet in an established virtual circuit

phase. Router Rl forwards the packet to router R3, but changes the label to 66.Router R3I forwards the packet to router R4, but changes the label to 22. Finally, router R4 delivers the packet to its final destination with the label 77. All the packets in the message follow the same sequence of labels, and the packets arrive in order at the destination. *Teardown Phase* In the teardown phase, source A, after sending all packets to B, sends a special packet called a teardown packet. Destination B responds with a confirmation packet. All routers delete the corresponding entries from their tables.

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 an incoming packet should be transmitted on. If the network uses datagrams 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 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 think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use 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 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 and also 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
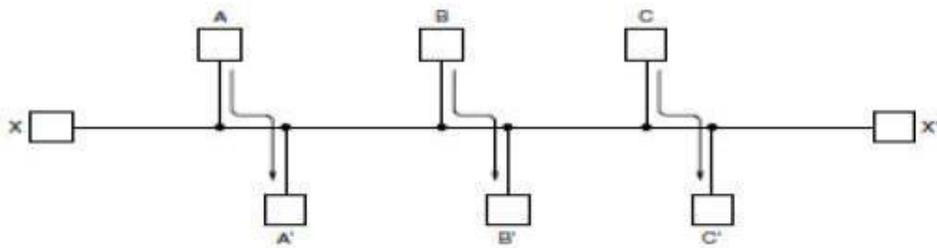
Fig: Network with a conflict between fairness and efficiency.

and traffic. 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 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 $r2$. If a route better than $r2$ existed from J to K, it could be concatenated with $r1$ to improve the route from I to K, contradicting our statement that $r1r2$ 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.
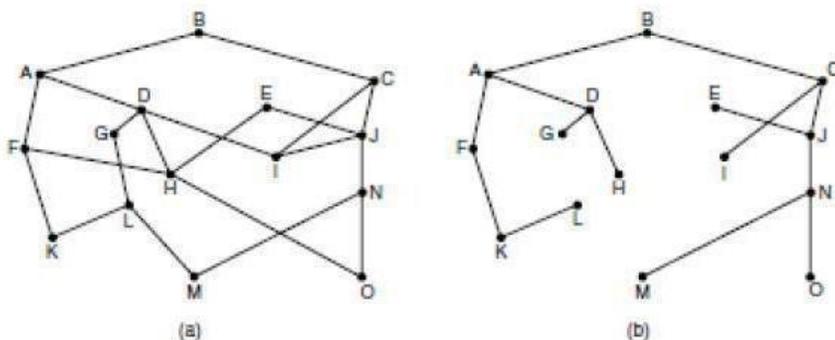
Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a **DAG** (**Directed Acyclic Graph**). DAGs have no loops. We will use sink trees as convenient shorthand 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 back up 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).
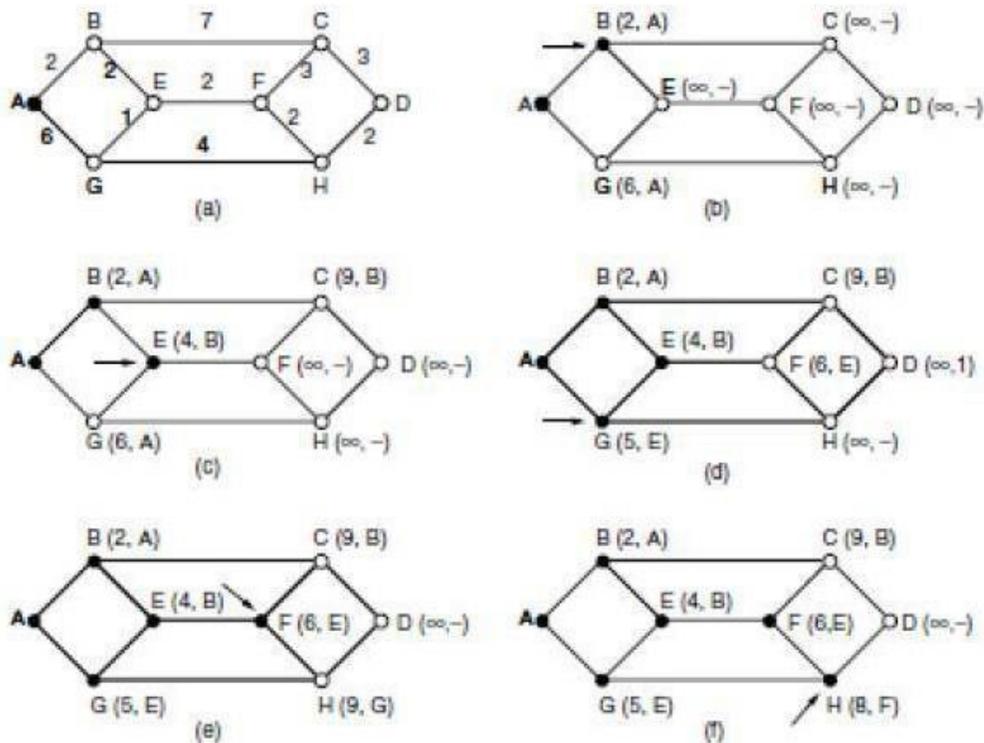
The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

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, each of 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 that of *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 same as 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.

Dijkstra's algorithm to compute the shortest path through a graph

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 produce a 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 **Bellman-Ford** 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.

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

Vectors received from
J's four neighbors

(a)   (b)
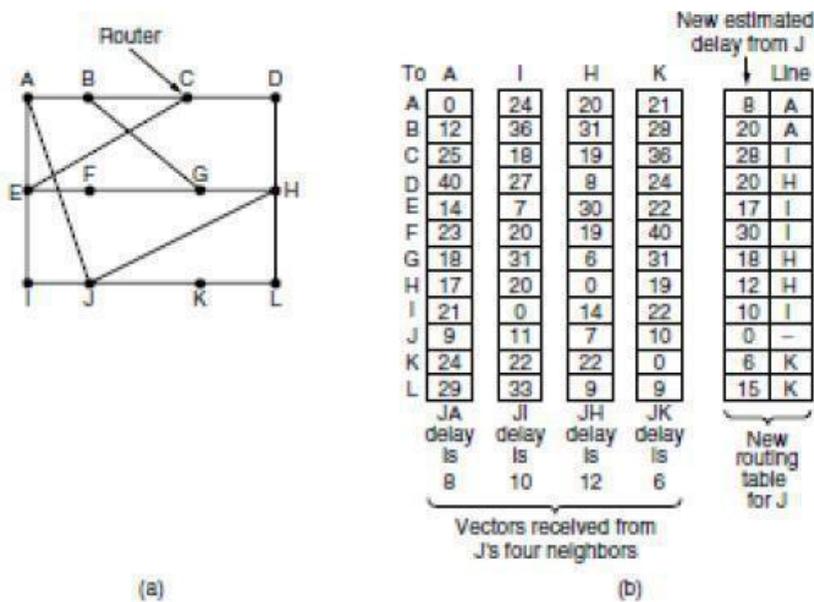
The Count-to-Infinity Problem

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

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

(a)

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

(b)
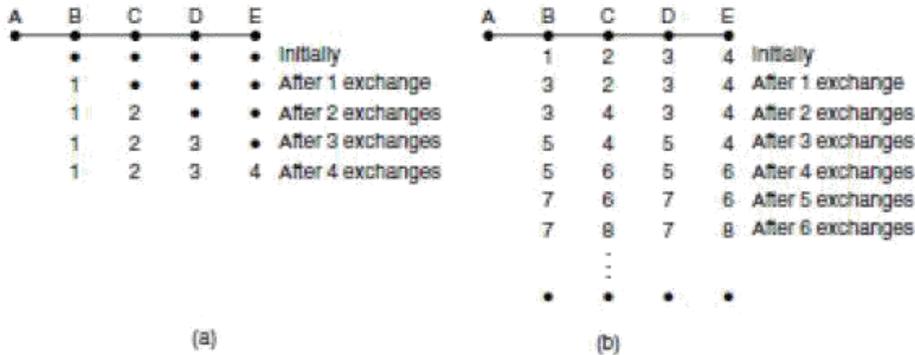
number hops. Suppose *A* is down initially and all the other routers know this. In other words, they have all recorded the delay to *A* as infinity.

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 left-hand 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.

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 to send 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 work its 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 of routers 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) discovered that 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 usuallyacceptable.

Hierarchical routing.

Full table for 1A

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

Hierarchical table for 1A

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

(a)   (b)   (c)
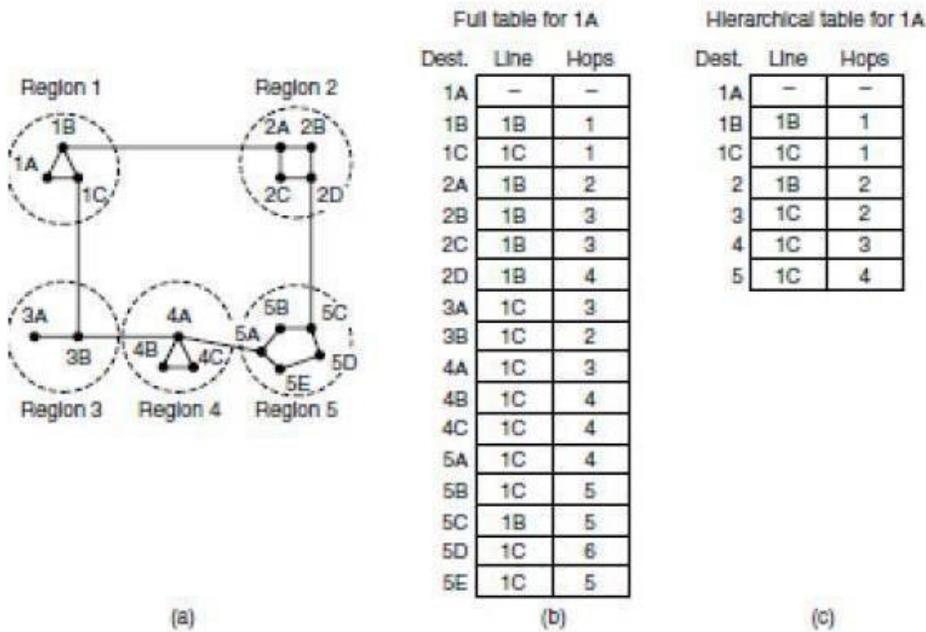
CONGESTION CONTROL ALGORITHMS

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 now congested.

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 the time 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 to a 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 trying to transfer files at 100 kbps to the other half. Here, the problem is not that of fast senders overpowering slow receivers, but that 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.

Algorithms

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.
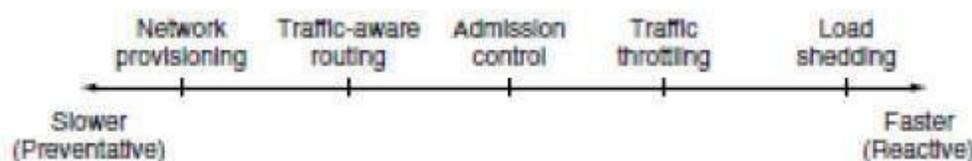
Timescales of approaches to congestion control.

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 slow down. 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 30minutes 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.

Traffic-Aware Routing

The first approach we will examine is traffic-aware routing. The routing schemes we looked at in Sec used fixed link weights. These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion. The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth

and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal. Traffic-aware routing was used in the early Internet according to this model (Khanna and Zinky, 1989). However, there is a peril. Consider the network of Fig., which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queuing delay in the weight used for the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, loading this link. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.
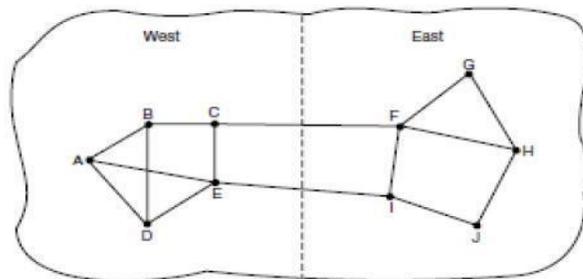
A network in which the East and West parts are connected by two links.

If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multipath routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge, as in the scheme of Gallagher (1977).Given these difficulties, in the Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called **traffic engineering**.

Admission Control

One technique that is widely used in virtual-circuit networks to keep congestion at bay is
admission control. The idea is simple: do not set up a new virtual circuit unless the network can
carry the added traffic without becoming congested. Thus, attempts to set up a virtual circuit may
fail. This is better than the alternative, as letting more people in when the network is busy just
makes matters worse. By analogy, in the telephone system, when a switch gets overloaded it
practices admission control by not giving dial tones. The trick with this approach is working out
when a new virtual circuit will lead to congestion. The task is straightforward in the telephone
network because of the fixed bandwidth of calls (64 kbps for uncompressed audio). However,
virtual circuits in computer networks come in all shapes and sizes. Thus, the circuit must come
with some characterization of its traffic if we are to apply admission control. Traffic is often
described in terms of its rate and shape.

The problem of how to describe it in a simple yet meaningful way is difficult because
traffic is typically bursty—the average rate is only half the story. For example, traffic that varies
while browsing the Web is more difficult to handle than a streaming movie with the same long-
term throughput because the bursts of Web traffic are more likely to congest routers in the
network. A commonly used descriptor that captures this effect is the **leaky bucket** or **token
bucket**. A leaky bucket has two parameters that bound the average rate and the instantaneous
burst size of traffic. Since leaky buckets are widely used for quality of service, we will go over
them in detail in Sec. Armed with traffic descriptions, the network can decide whether to admit
the new virtual circuit. One possibility is for the network to reserve enough capacity along the
paths of each of its virtual circuits that congestion will not occur. In this case, the traffic
description is a service agreement for what the network will guarantee its users. We have
prevented congestion but veered into the related topic of quality of service a little too early; we
will return to it in the next section. Even without making guarantees, the network can use traffic
descriptions for admission control. The task is then to estimate how many circuits will fit within
the carrying capacity of the network without congestion. Suppose that virtual circuits that may
blast traffic at rates up to 10 Mbps all pass through the same 100-Mbps physical link. How many
circuits should be admitted? Clearly, 10 circuits can be admitted without risking congestion, but
this is wasteful in the normal case since it may rarely happen that all 10 are transmitting full blast
at the same time. In real networks, measurements of past behavior that capture the statistics of

transmissions can be used to estimate the number of circuits to admit, to trade better performance for acceptable risk. Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig (a), in which two routers are congested, as indicated.

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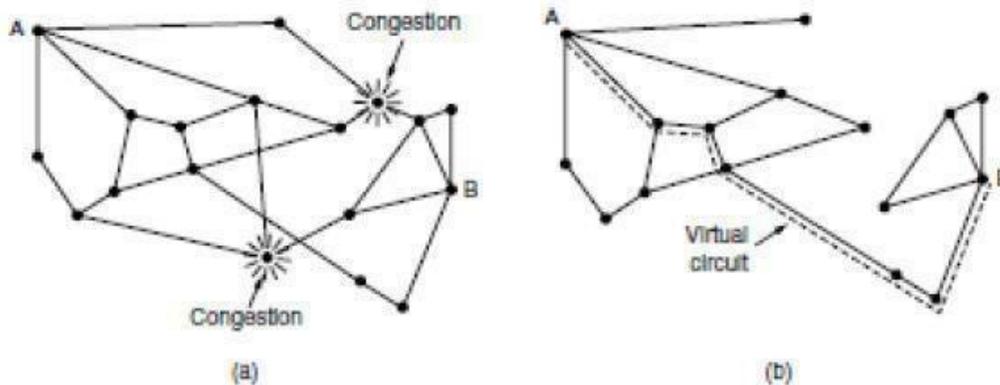
A congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

Suppose that a host attached to router *A* wants to set up a connection to a host attached to router *B*. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the network as shown in Fig. 5-24(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers. Shaikh et al. (1999) give a design for this kind of load-sensitive routing.

IPV4 Addresses

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 and top to bottom, with the high-order bit of the *Version* field going first. (This is a ''big-endian'' network byte order. On little endian 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 to dominate computing.
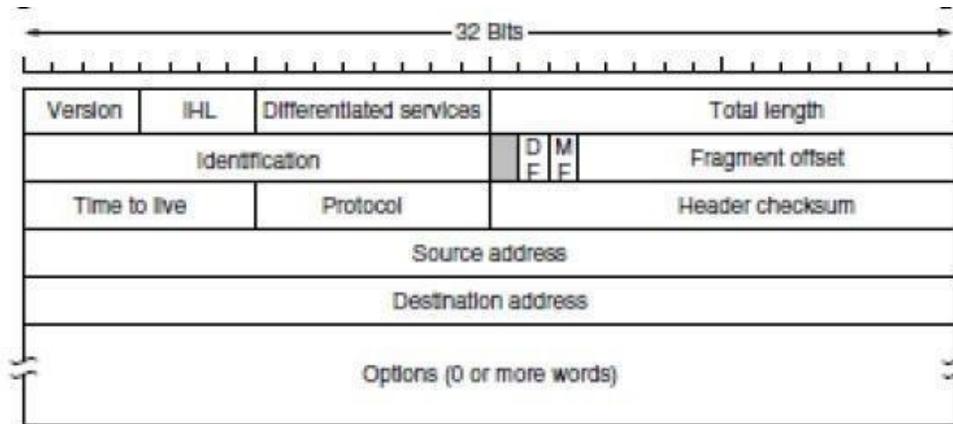
The IPv4 (Internet Protocol) header.

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, or an 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. Working together, the *Identification*, *MF*, and *Fragment offset* fields are used to implement fragmentation as described in Sec. 5.5.5.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 half words 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. 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. were defined.

The *Security* option tells how secret the information is. In theory, a military router might use this field to specify not to route packets through certain countries the military considers to be ''bad guys.'' In practice, all routers ignore it, so its only practical function is to help spies find

the good stuff more easily. The *Strict source routing* option gives the complete path from source to destination as a sequence of IP addresses. The datagram is required to follow that

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Some of the IP options exact route. It is most useful for system managers who need to send emergency packets when the routing tables have been corrupted, or for making timing measurements. The *Loose source routing* option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. Normally, this option will provide only a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.

The *Record route* option tells each router along the path to append its IP address to the *Options* field. This allows system managers to track down bugs in the routing algorithms (''Why are packets from Houston to Dallas visiting Tokyo first?''). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of options was plenty. As mentioned above, now it is too small. Finally, the *Timestamp* option is like the *Record route* option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly useful for network measurement. Today, IP options have fallen out of favor. Many routers ignore them or do not process them efficiently, shunting them to the side as an uncommon case. That is, they are only partly supported and they are rarely used.

Connecting Devices

Devices that are used to connect other devices are called connecting devices. That is, connecting device is a device that connects.

There are 4 types of connecting devices. They are as follows,

i. Repeater:

A repeater is a device that receives signals and before it become too weak or corrupted the repeater regenerates the original bit pattern and sends the data. It is like a refreshing station for data while travelling. Some properties are as follows,
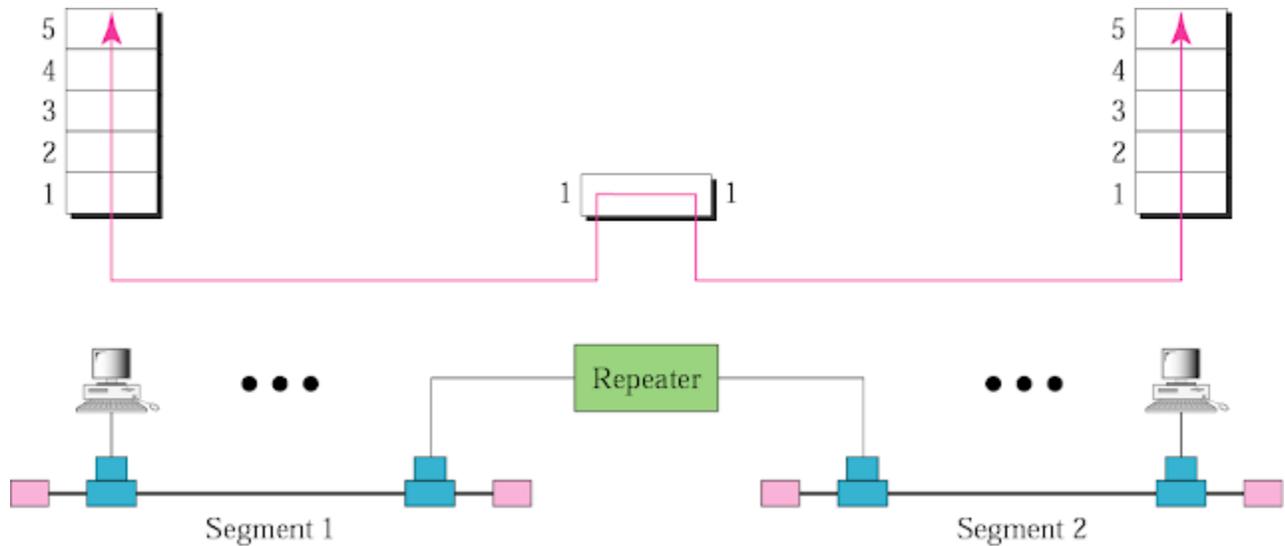
Segment 1   Segment 2

1. It operates only in the physical layer.
2. It connects two segments of the same LAN.
3. It forwards every bits it receives.
4. It regenerate bits, does not amplify them.

ii. Hub:

A hub is a multi-port repeater. That is a hub can connect more than two segments of a LAN where a repeater can connect only two segments. It also does not have any filtering capabilities like repeater.
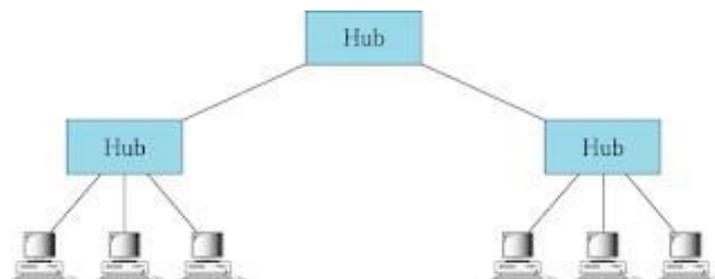
iii. Bridge:

A bridge operates in both physical layer and data link layer. As a physical layer device, it can regenerate bits and as a data link layer device it can check the physical addresses contained in the frame. Bits can be carried to any distance by using properly spaced repeaters but the collision domain will not allow this and that's why brides are required.

Address	Interface
712B13456141	1
712B13456142	1
642B13456112	2
642B13456113	2
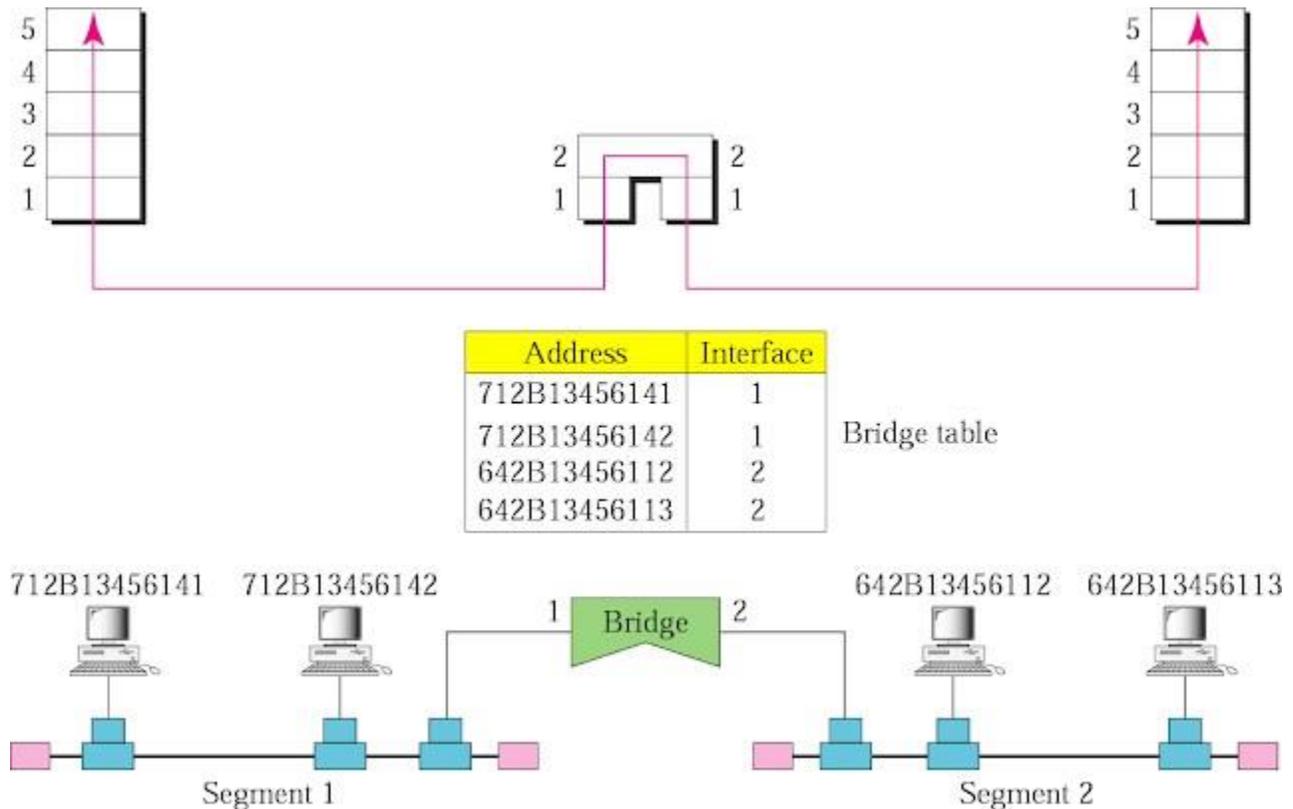
Bridge table

Bridge has filtering capability. It has table that maps addresses to port. A bridge can check the table to decide if a frame has to be forwarded or dropped. Bridges also work in a single LAN. There are some bridges called transparent bridge which can generate its address table automatically by learning the frame movement in the network.

iv. Router:

A router is three layer device; physical, data link and network layer. As a physical layer device it can regenerates bit, as data link layer device the router can check the physical addresses contained in a packet and as a network layer device it can check the network layer addresses (addresses in IP layer).
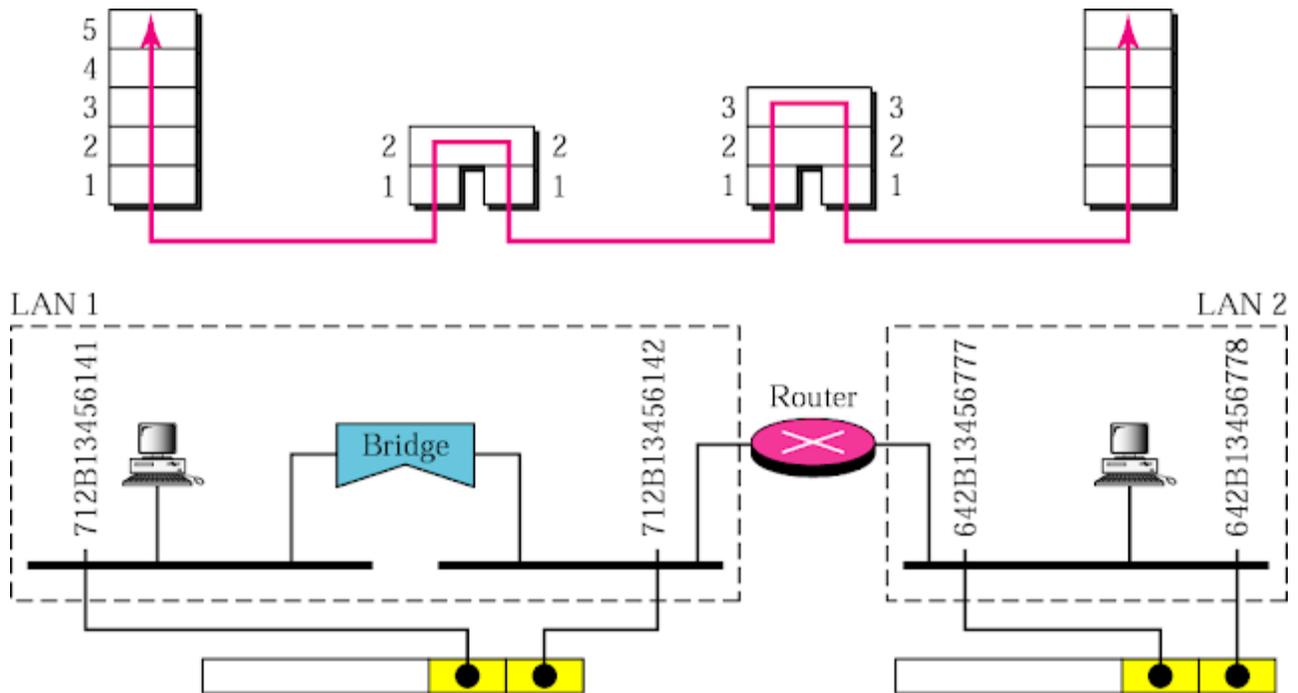
Routers are inter-networking devices as it can connects LAN-LAN, LAN-WAN, WAN-WAN. Routers can change the physical addresses of a packet. For example, considering a packet is getting sent from LAN 1 to LAN 2 as in the above given figure. At this moment the source address is of the sender's address and the destination address is of a host of the LAN 2. Now when the packet reaches the router, the router changes the source address of the packet to its own address and then send the packet to the destination address.

Virtual LAN IP Version 6 Addresses

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 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? In addition to the address problems, other issues loom in the background. In its early years, the Internet was largely used by universities, high-tech industries, and the U.S. Government (especially the Dept. of Defense). With the explosion of interest in the Internet starting in the mid-1990s, it began to be used by a different group of people, often with different requirements. For one thing, numerous people with smart phones use it to keep in contact with their home bases. For another, with the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, resulting in a billion machines being used for audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible. Seeing these problems on the horizon, in 1990 IETF started work on a new version of IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

- Support billions of hosts, even with inefficient address allocation.
- Reduce the size of the routing tables.
- Simplify the protocol, to allow routers to process packets faster.
- Provide better security (authentication and privacy).
- Pay more attention to the type of service, particularly for real-time data.
- Aid multicasting by allowing scopes to be specified.
- Make it possible for a host to roam without changing its address.
- Allow the protocol to evolve in the future.
- Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. To develop a protocol that met all these requirements, IETF issued

a call for proposals and discussion in RFC 1550. Twenty-one responses were initially received. By December 1992, seven serious proposals were on the table. They ranged from making minor patches to IP, to throwing it out altogether and replacing it with a completely different protocol. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever as it could give every molecule of water in the oceans enough addresses (roughly 25) to set up a small network. This choice would also have unified two major network layer protocols. However, many people felt that this would have been an admission that something in the OSI world was actually done right, a statement considered Politically Incorrect in Internet circles. CLNP was patterned closely on IP, so the two are not really that different. In fact, the protocol ultimately chosen differs from IP far more than CLNP does. Another strike against CLNP was its poor support for service types, something required to transmit multimedia efficiently.

Three of the better proposals were published in *IEEE Network* (Deering, 1993; Francis, 1993; and Katz and Ford, 1993). After much discussion, revision, and jockeying for position, a modified combined version of the Deering and Francis proposals, by now called **SIPP** (**Simple Internet Protocol Plus**) was selected and given the designation **IPv6.** IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses. The main features of IPv6 are discussed below. More information about it can be found in RFCs 2460 through 2466.First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses. We will have more to say about addresses shortly. The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay. We will discuss the header shortly, too. The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

A fourth area in which IPv6 represents a big advance is in security. IETF had its fill of newspaper stories about precocious 12-year-olds using their personal computers to break into banks and military bases all over the Internet. There was a strong feeling that something had to be done to improve security. Authentication and privacy are key features of the new IP. These were later retrofitted to IPv4, however, so in the area of security the differences are not so great any more. Finally, more attention has been paid to quality of service. Various halfhearted efforts to improve QoS have been made in the past, but now, with the growth of multimedia on the Internet, the sense of urgency is greater.

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 de-multiplexing, 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.
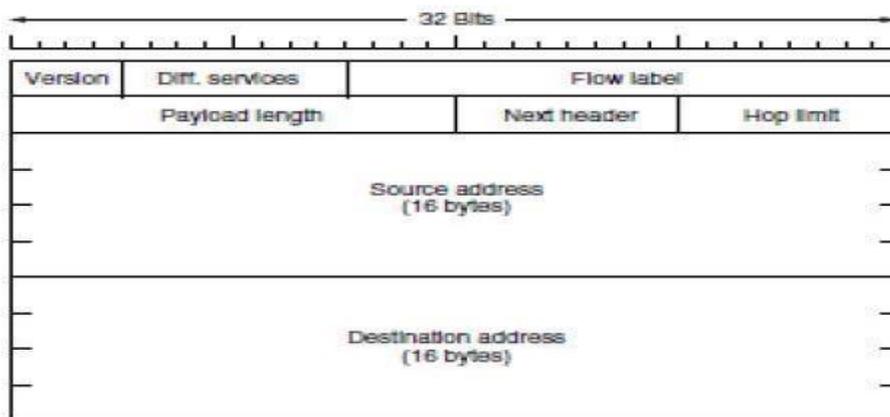
Fig: **The IPv6 fixed header (required).**

The **Differentiated services** field (originally called *Traffic class*) is used to distinguish the class of service for packets with different real-time delivery requirements. It is used with the

Differentiated service architecture for quality of service in the same manner as the field of the same name in the IPv4 packet. Also, the low-order 2 bits are used to signal explicit congestion indications, again in the same way as with IPv4.

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo connection. For example, a stream of packets from one process on a certain source host to a process on a specific destination host might have stringent delay requirements and thus need reserved bandwidth. The flow can be set up in advance and given an identifier. When a packet with a nonzero *Flow label* shows up, all the routers can look it up in internal tables to see what kind of special treatment it requires. In effect, flows are an attempt to have it both ways: the flexibility of a datagram network and the guarantees of a virtual-circuit network. Each flow for quality of service purposes is designated by the source address, destination address, and flow number.

This design means that up to 220 flows may be active at the same time between a given pair of IP addresses. It also means that even if two flows coming from different hosts but with the same flow label pass through the same router, the router will be able to tell them apart using the source and destination addresses. It is expected that flow labels will be chosen randomly, rather than assigned sequentially starting at 1, so routers are expected to hash them. The *Payload length* field tells how many bytes follow the 40-byte header of Fig. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one.

If this header is the last IP header, the *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to. The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In theory, in IPv4 it was a time in seconds, but no router used it that way, so the name was changed to reflect the way it is actually used. Next come the *Source address* and *Destination address* fields. Deering's original proposal, SIP, used 8-byte addresses,

but during the review process many people felt that with 8-byte addresses IPv6 would run out of addresses within a few decades, whereas with 16-byte addresses it would never run out. Other people argued that 16 bytes was overkill, whereas still others favored using 20-byte addresses to be compatible with the OSI datagram protocol. Still another faction wanted variable-sized addresses. After much debate and more than a few words unprintable in an academic textbook, it was decided that fixed-length 16-byte addresses were the best compromise.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this: 8000:0000:0000:0000:0123:4567:89AB: CDEF Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes 8000::123:4567:89AB:CDEF Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number,

For example: ::192.31.20.46 Perhaps it is unnecessary to be so explicit about it, but there are a lot of 16- addresses per square meter. Students of chemistry will notice that this number is larger than Avogadro's number. While it was not the intention to give every molecule on the surface of the earth its own IP address, we are not that far off. In practice, the address space will not be used efficiently, just as the telephone number address space is not (the area code for Manhattan, 212, is nearly full, but that for Wyoming, 307, is nearly empty). In RFC 3194, Durand and Huitema calculated that, using the allocation of telephone numbers as a guide, even in the most pessimistic scenario there will still be well over 1000 IP addresses per square meter of the entire earth's surface (land and water). In any likely scenario, there will be trillions of them per square meter. In short, it seems unlikely that we will run out in the foreseeable future. It is instructive to compare the IPv4 header (Fig.) with the IPv6 header (Fig) to see what has been left out in IPv6. The *IHL* field is gone because the IPv6 header has a fixed length. The *Protocol* field was taken out because the *Next header* field tells what follows the last IP header (e.g., a UDP or TCP segment).All the fields relating to fragmentation were removed because IPv6 takes a different approach to fragmentation. To start with, all IPv6-conformant hosts are expected to dynamically determine the packet size to use. They do this using the path MTU discovery procedure we described in Sec. 5.5.5. In brief, when a host sends an IPv6 packet that is too large, instead of fragmenting it, the router that is unable to forward it drops the packet and sends an error message back to the sending host. This message tells the host to break up all future packets

to that destination. Having the host send packets that are the right size in the first place is ultimately much more efficient than having the routers fragment them on the fly. Also, the minimum-size packet that routers must be able to forward has been raised from 576 to 1280 bytes to allow 1024 bytes of data and many headers. Finally, the *Checksum* field is gone because calculating it greatly reduces performance. With the reliable networks now used, combined with the fact that the data link layer and transport layers normally have their own checksums, the value of yet another checksum was deemed not worth the performance price it extracted. Removing all these features has resulted in a lean and mean network layer protocol. Thus, the goal of IPv6—a fast, yet flexible, protocol with plenty of address space—is met by this design.

Extension Headers

Some of the missing IPv4 fields are occasionally still needed, so IPv6 introduces the concept of (optional) **extension headers**. These headers can be supplied to provide extra information, but encoded in an efficient way. Six kinds of extension headers are defined at present, as listed in Fig. Each one is optional, but if more than one is present they must appear directly after the fixed header, and preferably in the order listed.

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

IPv6 extension headers

Some of the headers have a fixed format; others contain a variable number of variable-length options. For these, each item is encoded as a (*Type, Length, Value*) tuple. The *Type* is a 1-byte field telling which option this is. The *Type* values have been chosen so that the first 2 bits tell routers that do not know how to process the option what to do. The choices are: skip the option; discard the packet; discard the packet and send back an ICMP packet; and discard the packet but do not send ICMP packets for multicast addresses (to prevent one bad multicast packet from generating millions of ICMP reports). The *Length* is also a 1-byte field. It tells how long the value is (0 to 255 bytes). The *Value* is any information required, up to 255 bytes. The hop-by-hop header is used for information that all routers along the path must examine. So far, one option has been defined: support of datagrams exceeding 64 KB. The format of this header is shown in. When it is used, the *Payload length* field in the fixed header is set to 0.

Next header	0	194	4
Jumbo payload length			
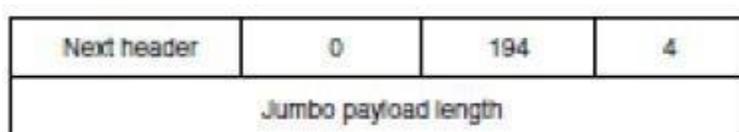
The hop-by-hop extension header for large data grams (jumbo grams)

As with all extension headers, this one starts with a byte telling what kind of header comes next. This byte is followed by one telling how long the hop-by-hop header is in bytes, excluding the first 8 bytes, which are mandatory. All extensions begin this way. The next 2 bytes indicate that this option defines the datagram size (code 194) and that the size is a 4-byte number. The last 4 bytes give the size of the datagram. Sizes less than 65,536 bytes are not permitted and will result in the first router discarding the packet and sending back an ICMP error message. Datagrams using this header extension are called **jumbo grams**. The use of jumbo grams is important for supercomputer applications that must transfer gigabytes of data efficiently across the Internet.

The destination options header is intended for fields that need only be interpreted at the destination host. In the initial version of IPv6, the only options defined are null options for padding this header out to a multiple of 8 bytes, so initially it will not be used. It was included to make sure that new routing and host software can handle it, in case someone thinks of a destination option some day. The routing header lists one or more routers that must be visited on the way to the destination. It is very similar to the IPv4 loose source routing in that all addresses listed must be visited in order, but other routers not listed may be visited in between. The format of the routing header is shown in Fig.
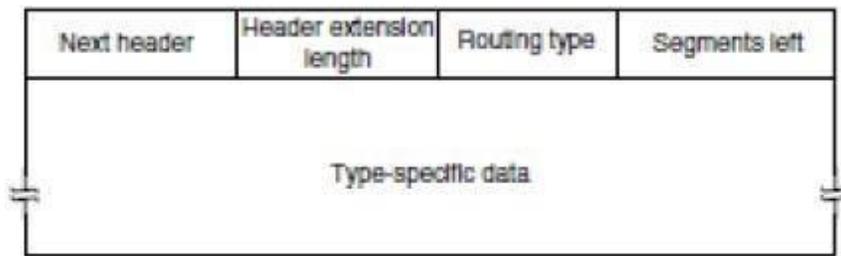
The extension header for routing

The first 4 bytes of the routing extension header contain four 1-byte integers. The *Next header* and *Header extension length* fields were described above. The *Routing type* field gives the format of the rest of the header. Type 0 says that a reserved 32-bit word follows the first word, followed by some number of IPv6 addresses. Other types may be invented in the future, as needed. Finally, the *Segments left* field keeps track of how many of the addresses in the list have not yet been visited. It is decremented every time one is visited. When it hits 0, the packet is on its own with no more guidance about what route to follow. Usually, at this point it is so close to the destination that the best route is obvious.

Internet Protocol

The **Internet Protocol** (**IP**) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

Historically, IP was the connectionless datagram service in the original Transmission Control Program introduced by Vint Cerf and Bob Kahn in 1974; the other being the connection-oriented Transmission Control Protocol (TCP). The Internet protocol suite is therefore often referred to as TCP/IP.

The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor, Internet Protocol Version 6 IPv6, has been growing in adoption for the last years, reaching almost 20% of the Internet traffic as of April, 2018

Hardware Addressing versus IP Addressing

Comparison Chart

BASIS FOR COMPARISON	MAC	IP
Full Form	Media Access Control Address.	Internet Protocol Address.
Purpose	It identifies the physical address of a computer on the internet.	It identifies connection of a computer on the internet.
Bits	It is 48 bits (6 bytes) hexadecimal address.	IPv4 is a 32-bit (4 bytes) address, and IPv6 is a 128-bits (16 bytes) address.
Address	MAC address is assigned by the manufacturer of NIC card.	IP address is assigned by the network administrator or Internet Service Provider.
Retrieve Address	ARP protocol can retrieve MAC address of a device.	RARP protocol can retrieve IP address of a device.

IP Data Gram

The IP datagram header format

Version

- Which version of the protocol the datagram belongs to.
- The current version number is 4.
- Next version: 6

IHL

- The number of 32-bit words in the header
- Because this is 4 bits, the max header length is 15 words (i.e. 60 bytes)
- The header is at least 20 bytes, but options may make it bigger

Type of Service

- Contains a 3-bit precedence field (that is ignored today), 4 service bits, and 1 unused bit.
- The four service bits can be:
  - 1000 - minimize delay
  - 0100 - maximize throughput
  - 0010 - maximize reliability
  - 0001 - minimize monetary cost
- This is a "hint" of what characteristics of the physical layer to use
- The Type of Service is not supported in most. implementations. However, some implementations have extra fields in the routing table to indicate delay, throughput, reliability, and monitary cost.

Total Length

- total length of the datagram in bytes.
- we know where the data starts by the header length
- we know the size of the data by computing "total length - header length"

Identification

- Uniquely identifies the datagram.
- Usually incremented by 1 each time a datagram is sent.
- All fragments of a datagram contain the same identification value.
- This allows the destination host to determine which fragment belongs to which datagram.

Flags and Fragmentation Offset

- Used for fragmentation
- DF means do not fragment. It is a request to routers not to fragment the datagram since the destination is incapable of putting the pieces back together.
- MF means more fragments to follow. All fragments except the last one have this bit set. It is needed to know if all fragments of a datagram have arrived.

Time to Live

- Upper limit of routers
- usually set to 32 or 64.
- decremented by each router that processes the datagram,
- router discards the datagram when TTL reaches 0.

Protocol

- Tells IP where to send the datagram up to.
- 6 means TCP
- 17 means UDP

UDP Protocol

UDP provides connectionless, unreliable, datagram service. Connectionless service means that there is no logical connection between the two ends exchanging messages. Each message is an independent entity encapsulated in a datagram.

UDP does not see any relation (connection) between consequent datagram coming from the same source and going to the same destination.

UDP has an advantage: it is message-oriented. It gives boundaries to the messages exchanged. An application program may be designed to use UDP if it is sending small messages and the simplicity and speed is more important for the application than reliability.

User Datagram

UDP packets, called *user datagram,* have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).

The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes. The last field can carry the optional checksum

User datagram packet format
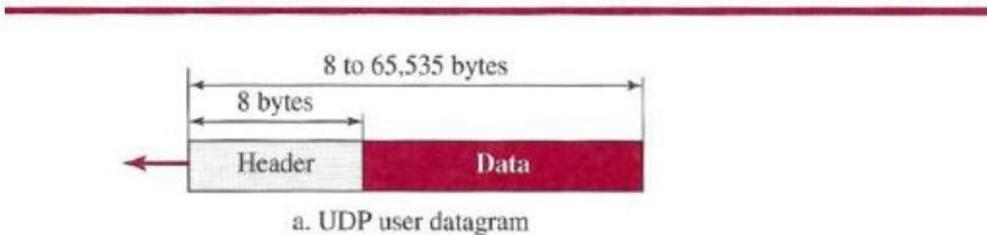
a. UDP user datagram

b. Header format

UDP Services

Process-to-Process Communication

UDP provides process-to-process communication using **socket addresses,** a combination of **IP** addresses and port numbers.

Connectionless Services

As mentioned previously, UDP provides a *connection less service.* This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user data grams even if they are coming from the same source process and going to the same destination program.

Flow Control

UDP is a very simple protocol. There is no *flow control,* and hence no window mechanism. The receiver may overflow with incoming messages.

Error Control

There is no *error control* mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.

Checksum

UDP checksum calculation includes three sections: a pseudo header, the UDP header, and the data coming from the application layer. The *pseudo header* is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s.

Pseudoheader for checksum calculation
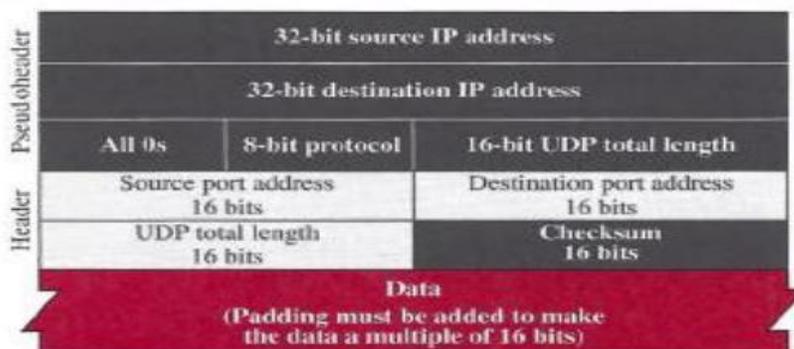
UDP Applications

UDP Features

Connectionless Service

As we mentioned previously,

- ➢ UDP is a connectionless protocol. Each UDP packet is independent from other packets sent by the same application program. This feature can be considered as an advantage or disadvanta8e depending on the application requirements.

- ➢ UDP does not provide error control; it provides an unreliable service. Most applications expect reliable service from a transport-layer protocol. Although a reliable service is desirable.

Typical Applications

- ➢ The following shows some typical applications that can benefit more from the services of UDP UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control

- ➢ UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFIP)

- ➢ UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software
- ➢ UDP is used for management processes such as SNMP

- ➢ UDP is used for some route updating protocols such as Routing Information Protocol (RIP)

- ➢ UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message

TCP Protocol

Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP Services

Process-to-Process Communication

As with UDP, TCP provides process-to-process communication using port numbers. We have already given some of the port numbers used by TCP.

Stream Delivery Service

In UDP, a process sends messages with predefined boundaries to UDP for delivery. UDP adds its own header to each of these messages and delivers it to IP for transmission.

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.

TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet.
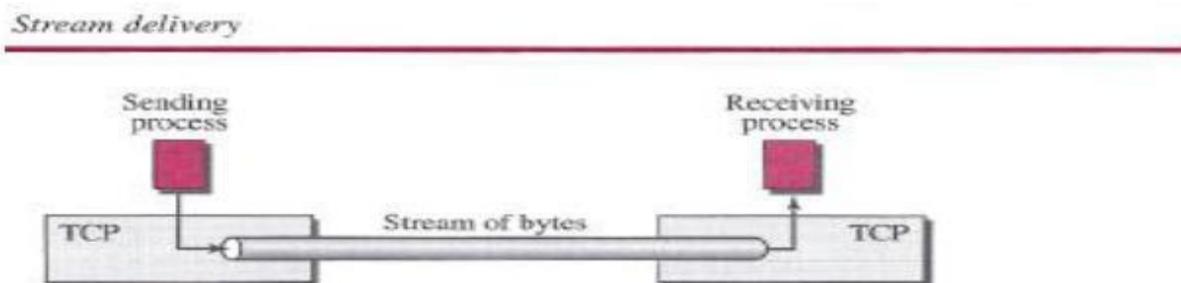
Sending and Receiving Buffers

Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage.

There are two buffers, the sending buffer and the receiving buffer, one for each direction.

> ➢ At the sender, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer).

> ➢ The colored area holds bytes that have been sent but not yet acknowledged.

> ➢ The TCP sender keeps these bytes in the buffer until it receives an acknowledgment. The shaded area contains bytes to be sent by the sending TCP.

> ➢ The operation of the buffer at the receiver is simpler. The circular buffer is divided into two areas (shown as white and colored).

> ➢ The white area contains empty chambers to be filled by bytes received from the network.

> ➢ The colored sections contain received bytes that can be read by the receiving process. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

Sending and receiving buffers
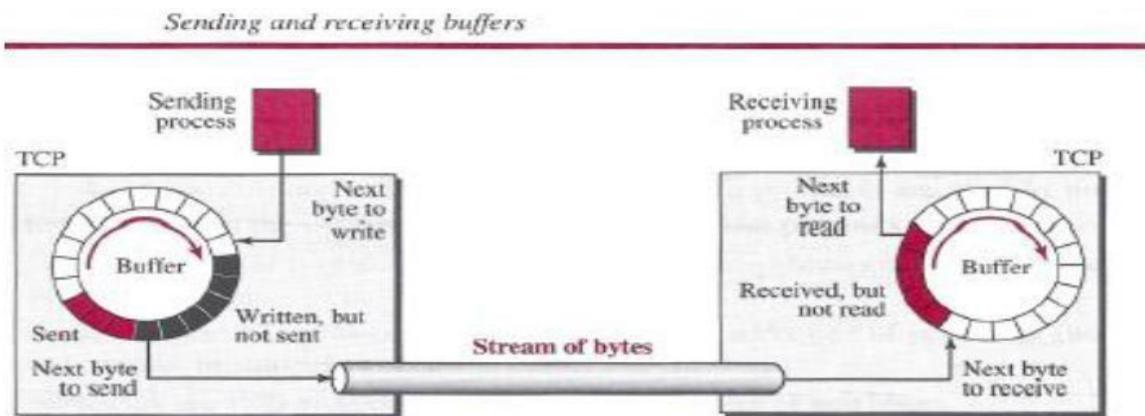
Segments

➢ Although buffering handles the disparity between the speed of the producing and consuming Processes, we need one more step before we can send data.

➢ The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a *segment.*

➢ The segments are encapsulated in an IP datagram and transmitted. This entire operation is transparent to the receiving process.

Format

The segment consists of a header of 20 to 60 bytes, followed by data from the application program.The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

Source port address This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

Destination port address This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

Sequence number This 32-bit field defines the number assigned to the first byte of data contained in this segment.

Acknowledgment number This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.

Header length This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.

A TCP Connection

➢ TCP is connection-oriented. a connection-oriented transport protocol establishes a logical path between the source and destination.

➢ All of the segments belonging to a message are then sent over this logical path.

➢ TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.

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

Connection Establishment

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.

Three- Way Handshaking

The connection establishment in TCP is called *three-way handshaking.* an application program, called the *client,* wants to make a connection with another application program, called the *server,*
using TCP as the transport-layer protocol The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This request is called a *passive open.*

　　Although the server TCP is ready to accept a connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP to connect to a particular server.

➢ A SYN segment cannot carry data, but it consumes one sequence number.
➢ A SYN + ACK segment cannot carry data, but it does consume one sequence number.
➢ An ACK segment, if carrying no data, consumes no sequence number.

Connection termination using three-way handshaking

The FIN + ACK segment consumes only one sequence number if it does not carry data.
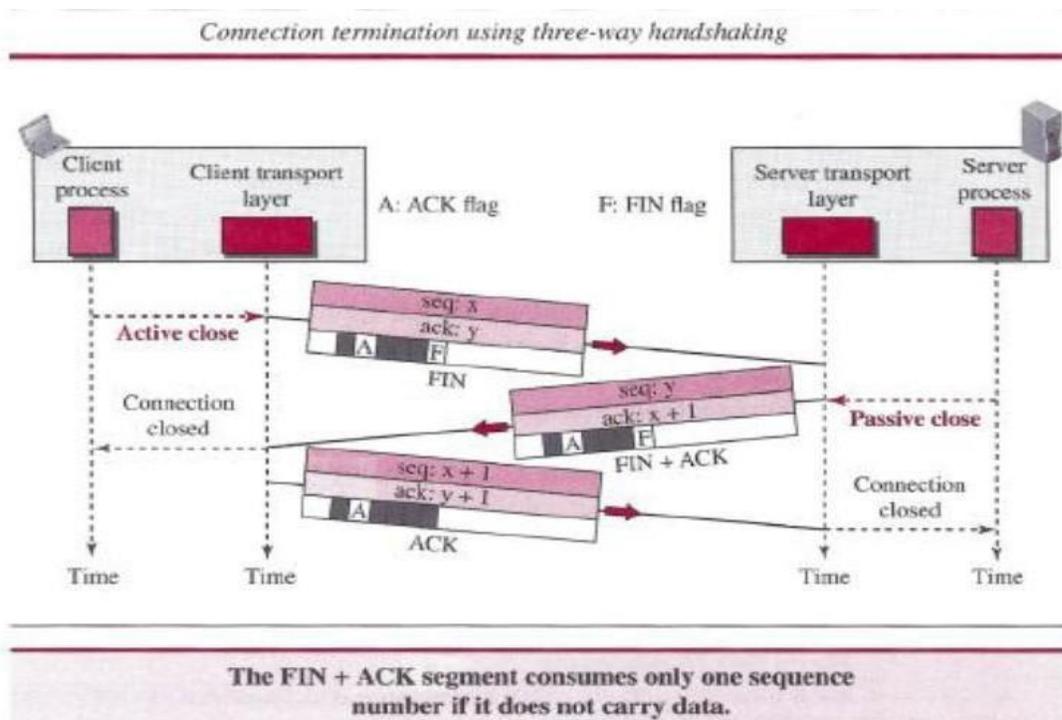
ATM

ATM is a form of packet switching technology. That is, ATM networks transmit their information in small, fixed length packets called "cell" each of which contains 48-octets (or bytes) of data and 5-octets of header information. The small, fixed cell size was chosen to facilitate the rapid processing of packet in hardware and to minimize the amount of time required to fill a single packet. This is particularly important for real-time applications such as voice and video that require short packetization delays. ATM is the transfer mode for implementing Broadband Integrated Service Digital Networks (B-ISDN) .

The term transfer comprises both transmission and switching aspects, so a transfer mode is a specific way of transmitting and switching information in a network. The term asynchronous, in new transfer mode name refers to the fact that, in the context of multiplexed transmission, cells allocated to the same connection may exhibit an irregular recurrence pattern as they are filled according to the actual demand, this is shown in Figure (a).

Cell

	channel 1		channel 5		channel 1		channel unused		channel 7		channel 5		channel 1
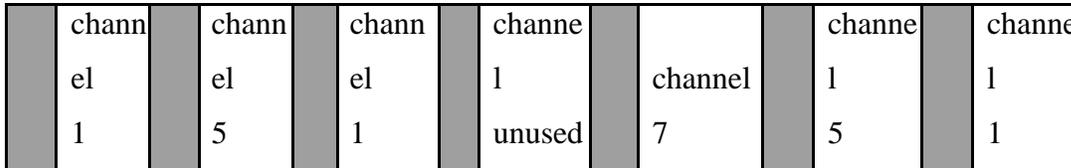
(a) Asynchronous Transfer Mode (ATM)

frame

	channel 1	channel 2	...	channel n		channel 1	channel 2	...	channel n		
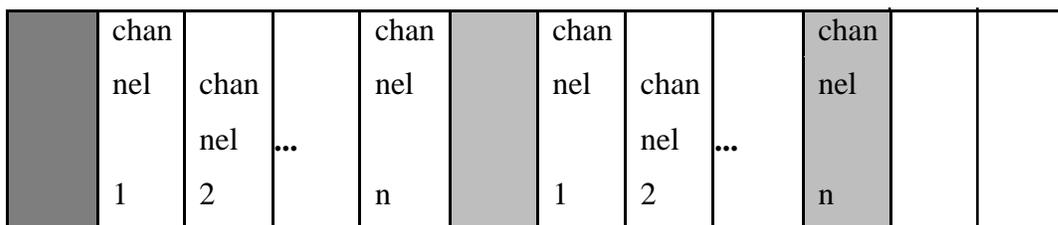
(b) Synchronous Transfer Mode (STM)

Framing signal

☐

User Information

▣

Header (contain routing identifier)

Fig: Synchronous and Asynchronous Transfer Modes .

Above Figure describes the difference between the Synchronous Transfer Mode (STM), and the Asynchronous Transfer Mode (ATM). As we have mentioned above that ATM is the data transfer interface for B-ISDN, let's give short notes about B-ISDN standards.

B-ISDN Standards

In 1988, the telecommunication standardization sector of the ITU, the international standards agency commissioned by the United Nations for the global standardization of telecommunications, has developed a number of standards for ATM netwo rks. Other standards bodies and consortia (e.g., the ATM Forum, ANSI) have also contributed to the development of ATM standards. The following subsection presents an overview of the standards, with particular emphasis on the protocol reference model used by ATM

Protocol Reference Model

The B-ISDN protocol reference model, defined in ITU-T recommendation I-321, is shown in Figure 2-2 The purpose of the protocol reference model is to clarify the functions that ATM networks perform by grouping them into a set of interrelated, function-specific layers and planes. The reference model consists of a user plane, a control plane and a management plane Within the user and control planes is a hierarchical set of layers. The user plane defines a set of functions for the transfer of user information between communication end-points; the control plane defines control functions such as call establishment, call maintenance, and call release, and the management plane defines the operations necessary to control information flow between planes and layers, and maintain accurate and fault - tolerant network operation.

Within the user and control planes, there are three layers; the physical layer, the ATM layer, and the ATM adaptation layer (AAL).

Table 1 summarizes the functions of each layer . The physical layer performs primarily bit level functions, the ATM layer is primarily responsible for the switching of ATM cells, and the ATM adaptation layer is responsible for the conversion of higher layer protocol forms into ATM cells. The function that the physical, ATM, and adaptation layers perform are described in more detail in the following:

	Higher Layer Functions		Higher Layers		
Layer managemen t	.convergence	CS	AAL		
	.segmentation and reassembly	SAR			
	.generic flow control .cell-header generation/extraction .cell VPI/VCI translation .cell multiplex and demultiplex		ATM layer		
	.cell-rate decoupling .HEC, header-sequence generation/verification .cell delineation .transmission -frame adaptation .transmission -frame generation/recovery	TC	physical layer	PHY independent	
	bit timing phy sical medium	PM		PHY dependent	

:

AAL : ATM Adaptation layer.   VCI   Virtual Channel Identifier.

CS   : Convergence Sublayer.   HEC : Header Error Control.

SAR : Segmentation And Reassembly.   TC   : Transmission Control.

VPI : Virtual Path Identifier.   PM : Physical Medium

Table 1 the Functions of B-ISDN in Relation to the B-ISDN PRM.

Physical Layer

The physical layer is divided into two sublayers: the physical medium sublayer, and the transmission convergence sublayer

Physical Medium (PM) sublayer

The physical medium sublayer performs medium-dependent functions. For example, it provides bit transmission capabilities including bit alignment, line coding and electrical/optical conversion. The PM sublayer is also responsible for bit timing, i.e., the insertion and extraction of bit timing information. The PM sublayer currently supports two types of interface: optical and electrical.

Transmission Convergence (TC) sublayer

Above the physical medium sublayer is the transmissio n convergence sublayer, which is primarily responsible for framing of data transported over the physical medium. The ITU-T recommendation specifies two options for TC sublayer transmission frame structure: cell-based and Synchronous Digital Hierarchy (SDH). In the cell-based case, cells are transported continuously without any regular frame structure. Under SDH, cells are carried in a special frame structure based on the North American SONET (Synchronous Optical Network) protocol Regardless of which transmission frame structure is used, the TC sublayer is responsible for the following four functions: cell rate decoupling, header error control, cell delineation, and transmission frame adaptation. Cell rate decoupling is the insertion of idle cells at the sending side to adapt the ATM cell stream's rate to the rate of the transmission path. Header error control is the insertion of an 8-bit CRC polynomial in the ATM cell header to protocol the contents of the ATM cell header. Cell delineation is the detection of cell boundaries. Transmission frame adaptation is the encapsulation of departing cells into an appropriate framing structure (either cell-based or SDH-based).

ATM Layer

The ATM layer lies a top the physical layer and specifies the functions required for the switching and flow control of ATM cells

There are two interfaces in an ATM network; the user-network-interface (UNI) between the ATM end -point and the ATM switch, and the network-network interface (NNI) between two ATM switches. Although a 48-octets cell payload is used at both interfaces, the 5-octets cell header differs slightly at these interfaces. Figure 2-3 shows the cell header structures used at the UNI and NNI

6. At the UNI, the header contains a 4-bits Generic Flow Control (GFC) field, a 24-bits label field containing VPI and VCI subfields (8-bits for the VPI and 16-bits for the VCI), a 2-bits payload type (PT) field, a 1-bit priority (PR) field, and an 8-bit header error check (HEC) field. The cell header for an NNI cell is identical to that for the UNI cell, except that it lacks the GFC field; these four bits are used for an additional 4 VPI bits in the NNI cell header.

8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
GFC / VPI	VPI
VPI / VCI	VPI | VCI
VCI	VCI
VCI | P T | CLP	VCI | PT | CLP
HEC	HEC
UNI format	NNI format

GFC : General Flow Control   PT: Payload Type.   CLP :   Cell   Loss
                                                       Priority
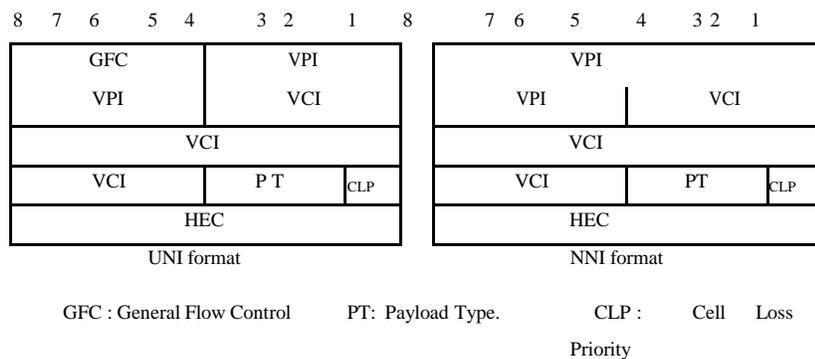
Figure 2-3 ATM Cell Header Format

The VCI and VPI fields are identifier values for VC and VP respectively. A virtual channel connects two ATM communication end-points. A virtual path connects two ATM devices, which can be switches or end-points.

Cryptography

Basic Concepts

Cryptography The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

Plaintext The original intelligible message

Cipher text The transformed message

Cipher An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

Key Some critical information used by the cipher, known only to the sender& receiver

Encipher (encode) The process of converting plaintext to cipher text using a cipher and a key

Decipher (decode) the process of converting cipher text back into plaintext using a cipher and a key

Cryptanalysis The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **code breaking**

Cryptology Both cryptography and cryptanalysis

Code An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Cryptography

Cryptographic systems are generally classified along 3 independent dimensions:

Type of operations used for transforming plain text to cipher text

All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.

The number of keys used

If the sender and receiver uses same key then it is said to be **symmetric key (or)**

single key (or) conventional encryption.

If the sender and receiver use different keys then it is said to be **public key encryption**.

The way in which the plain text is processed

A **block cipher** processes the input and block of elements at a time, producing output block for each input block.

A **stream cipher** processes the input elements continuously, producing output element one at a time, as it goes along.

Cryptanalysis

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

e.g., (i) the sequence of first letters of each word of the overall message spells out the real (Hidden) message.

(ii) Subset of the words of the overall message is used to convey the hidden message.

Various other techniques have been used historically, some of them are

Character marking – selected letters of printed or typewritten text are overwritten in pencil. The

marks are ordinarily not visible unless the paper is held to an angle to bright light.

Invisible ink – a number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light. Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography

Requires a lot of overhead to hide a relatively few bits of information. Once the system is discovered, it becomes virtually worthless.

SECURITY SERVICES

The classification of security services are as follows:

Confidentiality: Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.

E.g. Printing, displaying and other forms of disclosure.

Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

Access control: Requires that access to information resources may be controlled by or the target system.

Availability: Requires that computer system assets be available to authorized parties when needed.

NETWORK SECURITY

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1 **Encipherment**
2 **Digital Signature**
3 **Access Control**

SECURITY ATTACKS

There are four general categories of attack which are listed below.

Interruption

An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability e.g., destruction of piece of hardware, cutting of a communication line or Disabling of file management system.

Interception

An unauthorized party gains access to an asset. This is an attack on confidentiality. Unauthorized party could be a person, a program or a computer.e.g., wire tapping to capture data in the network, illicit copying of files

Sender   Receiver

Eavesdropper or forger
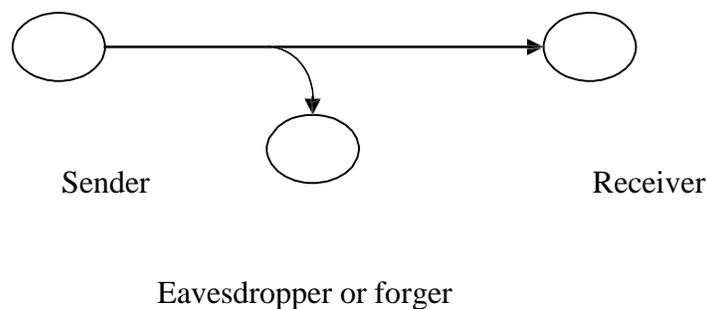
Modification

An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. e.g., changing values in data file, altering a program, modifying the contents of messages being transmitted in a network.
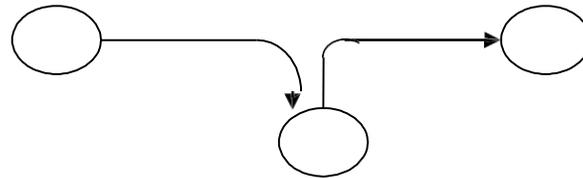
Sender   Receiver

Eavesdropper or forger

Fabrication

An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity. e.g., insertion of spurious message in a network or addition of records to a file.

Sender

Eavesdropper or forger