Unit I: Network Hardware: LAN – WAN – MAN – Wireless – Home Networks. Network Software: Protocol Hierarchies – Design Issues for the Layers – Connection-oriented and connectionless services – Service Primitives – The Relationship of services to Protocols. Reference Models: OSI Reference Model – TCP/IP reference Model – Comparison of OSI and TCP/IP -Critique of OSI and protocols – Critique of the TCP/IP Reference model.

OBJECTIVES

To learn about

- What is mean by network
- Different types of Networks
- What is the use of Layers
- Reference Models
- How to use Protocols in network.

COMPUTER NETWORKS

1.1 INTRODUCTION

- **USES OF COMPUTER NETWORKS**

1.1 INTRODUCTION

Computer Network means an “interconnected collection of autonomous Computers”. Two Computers are said to be interconnected if they can exchange information. The connection usually will be based on a communication medium like copper wire, fiber optics etc., Master/Slave relationship in which one computer forcibly start, stop, or control another computer is not a network, the computers should be autonomous.

Difference between a Computer Network and Distributed System

Computer Network	**Distributed System**
The Existence of autonomous computers are not transparent (they are visible).	The Existence of autonomous computers are transparent (they are not visible).
The autonomous computer performs the operation requested by the user.	The best processor is selected by the operating system for carrying out the operations requested by the user.
The user is aware of his working environment.	The user is not aware of his working environment, which is multiple processor in nature but looks like a virtual uniprocessor.
All operations (allocation of jobs to processors, files to disks, movement of files) are done explicitly.	All operations (allocation of jobs to processors, files to disks, movement of files) are done automatically without the user's knowledge.
Regulation software is enough for Computer networks.	A software that gives a high degree of cohesiveness and transparency is needed since distributed system is built on top of a network

1.2 USES OF COMPUTER NETWORKS

The use of Computer Networks can be divided into three ways.

1. Business Applications
2. Home Applications
3. Mobile Users
4. Social Issues

1.3 NETWORK HARDWARE

1.3.1 TRANSMISSION TECHNOLOGY

1.3.1 TRANSMISSION TECHNOLOGY

Broadly speaking, there are two types of transmission technology:

1. Broadcast networks.
2. Point-to-point Networks

1) BROADCAST NETWORKS

It has a single communication channel that is shared by all the machines on the network. Short messages, called **packets** in certain contexts sent by any machine are received by all the others. An address field within the packet specifies for whom it is intended. Upon receiving a packet, a machine checks the address field.

If the packet is intended for itself, it processes the packet; if the packet is intended for some other machine, it is just ignored. Although the packet may actually be received by many systems, only the intended one responds. The others just ignore it.

Broadcast systems also allow the possibility of addressing a packet to *all* destinations by using a special code in the address field. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called **broadcasting.** Some broadcast systems also support transmission to a subset of the machines, something known as **multicasting.**

One possible scheme is to reserve one bit to indicate multicasting. The remaining *n* - 1 address bits can hold a group number. Each machine can "subscribe" to any or all of the groups. When a packet is sent to a certain group, it is delivered to all machines subscribing to that group.

In contrast, **point-to-point** networks consist of many connections between individual pairs of machines. To go from the source to the destination, a packet on this type of network may have to first visit one or more intermediate machines.

Often multiple routes, of different lengths are possible, so routing algorithms play an important role in point-to-point networks. As a general rule smaller, geographically localized networks tend to use broadcasting, whereas larger networks usually are point-to-point. Point-to-Point transmission with one sender and one receiver is called **Unicasting.**An Interprocessor Distances Processors located in same Examples.

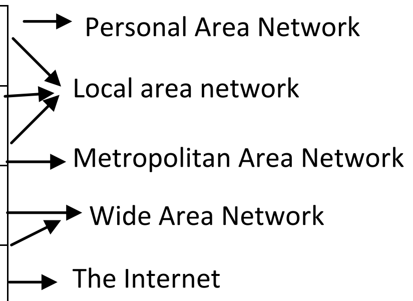
Multiple processor systems can be arranged by their physical size. At the top are **data flow machines,** highly parallel computers with many functional units all working on the same program. Next come the **multicomputers,** systems that communicate by sending messages over very short, very fast buses. Beyond the multicomputers are the true networks, computers that communicate by exchanging messages over longer cables. These can be divided into local, metropolitan, and wide area networks, finally, the connection of two or networks are called an internetwork. The worldwide Internet is a well-known example of an internet work.

1.3.2 There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important.

1. Transmission Technology
2. Scale

1.3.2 SCALE

1 m	Square meter
10 m	Room
100 m	Building
1 km	Campus
10 km	City
100 km	Country
1000 km	Continent
10000 km	Planet



Classification of interconnected processors by scale.

LOCAL AREA NETWORKS

Local area networks, generally called LANs, are privately-owned network within a single

building or campus of up to a few kilometers in size. They are widely used to connect personal computers and

workstations in company offices and factories to share resources (e.g., printers) and exchange information.

LANs are distinguished from other kinds of networks by three characteristics:

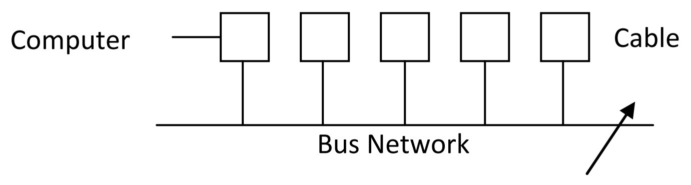
(1) Size.

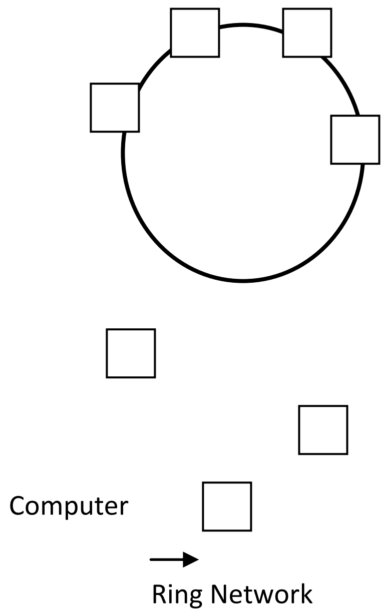
(2) Transmission Technology, and

(3) Topology.

LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. It simplifies network management. LANs often use a transmission technology consisting of a cable to which all the machines are attached. Traditional LANs run at speeds of 10 to *100* Mbps, have low delay (tens of microseconds), and make very few errors. Newer LANs may operate at higher speeds; up to hundreds of megabits/sec.

Various topologies are possible for broadcast LANs. In a bus (i.e., a linear cable) network, at any instant one machine is the Master and is allowed to transmit. All other machines are required to refrain from sending. An arbitration mechanism is needed to resolve conflicts when two or more machines want to transmit simultaneously.





The arbitration mechanism may be centralized or distributed. IEEE 802.3, popularly called Ethernet™, for example, is a bus-based broadcast network with decentralized control operating at 10 or 100 Mbps. Computers on an Ethernet can transmit whenever they want to; if two or more packets collide, each computer just waits a random time and tries again later.

A second type of broadcast system is the ring. In a ring, each bit propagates around on its own, not waiting for the rest of the packet to which it belongs. Typically, each bit circumnavigates the entire ring in the time it takes to transmit a few bits, often before the complete packet has even been transmitted. Like all other broadcast systems, some rule is needed for arbitrating simultaneous accesses to the ring. The IBM token ring, is a popular ring-based LAN operating at 4 and 16 Mbps.

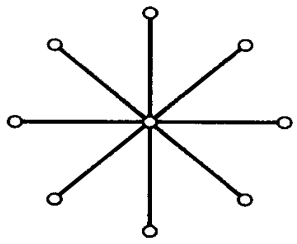
Broadcast networks can be further divided into static and dynamic, depending on how the channel is allocated. A typical static allocation would be to divide up time into discrete intervals and run a round robin algorithm, allowing each machine to broadcast only when its time slot comes up. Static allocation wastes channel capacity when a machine has nothing to say during its allocated slot, so most systems attempt to allocate the channel dynamically (i.e., on demand).

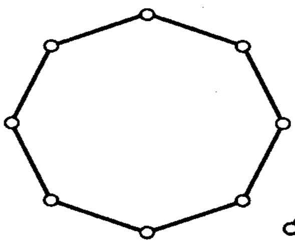
Dynamic allocation methods for a common channel are either centralized or decentralized. In the centralized

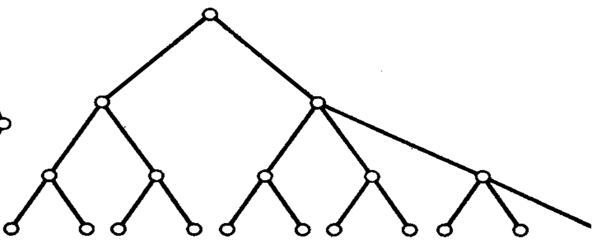
channel allocation method, there is a single entity, for example a bus arbitration unit, which determines who goes next. It might do this by accepting requests and making a decision according to some internal algorithm. In the decentralized channel allocation method, there *is* no central entity; each machine must decide for itself whether or not to transmit.

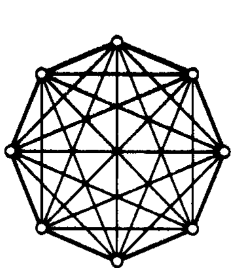
Topology for Point to Point Network

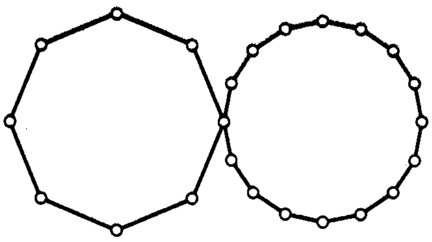
a) Star
b) Ring
c) Tree
d) Complete
e) Intersecting Rings
f) Irregular


(a)

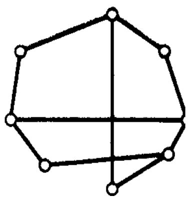

(b)


(c)


(d)


(e)

(f)



METROPOLITAN AREA NETWORKS

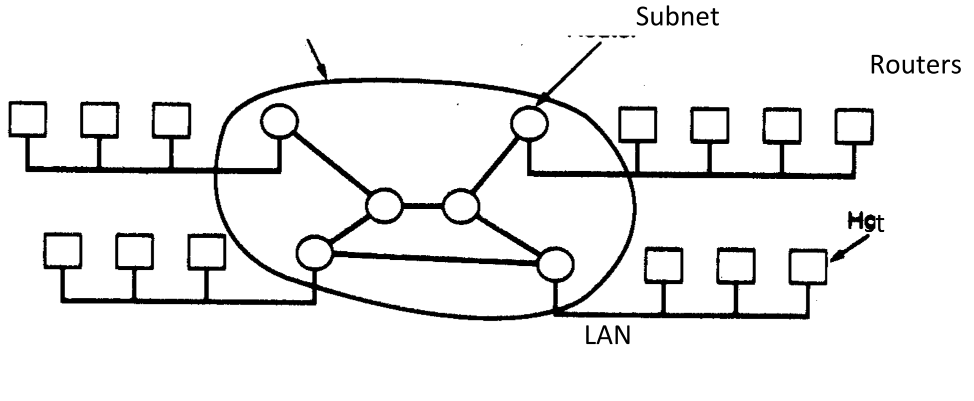
Metropolitan Area Networks or Man covers a city. The best-known example of MAN it is the cable television network available in many cities. In early systems a large antenna was placed on top of a nearby hill and signal was piped to the subscribers houses.

At first, these were locally-designed, ad hoc systems. The next step was television and even entire channels designed for cable only, starting when the internet attracted a mass audience a cable TV network operator begun to realize that with some changes to the system, they could provide two-way internet service in un used parts of the spectrum, we see both television signals and internet being fed into the centralized **head end** for subsequent distribution to homes.

WIDE AREA NETWORKS

Wide Area Networks or **WAN** spans a large geographical area often a country or continent. It contains collections of machines for running user programs called **Hosts.** The hosts are connected by a communication subnet. The hosts are owned by the customers whereas the communication subnet owned and operated by Telephone Company or ISP. The job of the subnet is to carry messages from host to host. The subnet consists of two distinct components: **transmission lines** and **switching elements**. **Transmission lines** move bits between machines that are made of copper wire, optical fiber, or even radio links. **Switching elements** are specialized computers that connect 3 or more transmission lines.

When data arrive on an incoming line the switching element must choose an outgoing line to forward them. The switching elements are also called router. The collection communication lines and routers(but not the hosts) form the subnet. A short subnet is a collection of communication lines that moved packets from the source host to the destination host. In most WAN the network contains numerous transmission lines, each one connecting a pair of routers. If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers.

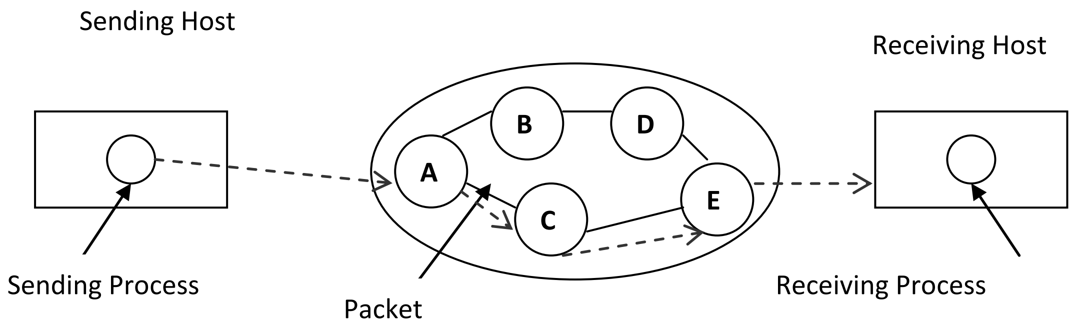


Relation between hosts and the subnet.

When a packet is sent from one router to another via one or more intermediate routers the packet is received at

each intermediate routers and stored there until the required line free and then forwarded. A subnet organized according to this principle is called store and forward or packet switched subnet. When the packets are small and all the same size they are often called as **cells.**

The principle of packet switched WAN, when a process on some host as a message to be sent to a Process on some other host the sending host first cuts the message into packets, each one bearing its number in the sequence. The packets are then transported individual over the network and deposited at the receiving hosts where they are reassembled into the original message and delivered to the receiving process.



A second possibility for a WAN is a satellite or ground radio system. Each router has an antenna through which it can send and receive. Sometimes the routers are connected to a substantial point-to-point subnet, with only some of them having a satellite antenna. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

WIRELESS NETWORKS

Wireless network can be divided into three main categories:

- System interconnection
- Wireless LANs
- Wireless WANs

1. System interconnection

System interconnection is all about interconnecting the components of a computer using short range radio. Every computer has monitor, keyboard, mouse and printer connected to main unit by cables. New users have a hard time plugging all the cables into right little holes. Some companies got together to design short range wireless network called **Bluetooth** to connect these components without wires. Bluetooth also allows digital cameras, headsets, Scanners and other devices to connect the computer about within range. No cables, no driver installation just put down them on and they work.

2. Wireless LANs

These are systems in which every computer has a radio modem and antenna which it can communicate with other systems. If systems are close enough they can communicate directly with one another with peer-to-peer configuration. Wireless LANs are becoming increasingly common in small offices and homes. There is a standard for wireless LANs called **IEEE 802.11.**

3. Wireless WANs

The radio network used for cellular telephones is an example of low bandwidth wireless systems. System has already gone through 3 generation. The first generation was analog and for voice only. The second generation was digital and for voice only. The third generation is digital and it is for both voice and data. Wireless LANs can operate rate up to 15 Mbps over distance of ten of meters. Cellular system operates below 10 Mbps but the distance between way station and the computer or telephone is measured in kilometers rather than meters. A standard for a called **IEEE 802.16**. For example an airplane with number peoples using modem and seat-back telephones to call the office. Each call is independent of other ones. Next case a flying LAN each seats comes equipped with an Ethernet connection into which passengers can plug their computers. A single router on the aircraft maintain radio link with some router on the ground, changing router as its flies along.

HOME NETWORK

Home network is on the horizon. The fundamental idea is that in the future most homes will be setup for networking. Every device in home will be capable of communicating with every other device, and all of them will be accessible over the internet. Many devices are capable of being a network some of more obvious categories are as follows:

Computers (desktop PC, notebook PC, PDA, shared peripherals)

- Entertainment (TV, DVD, VCR, Camcorder, camera, stereo, MP3)
- Telecommunication (telephone, mobile telephone, intercom, fax)
- Appliances (microwave, refrigerator, clock, furnace, airco, lights)
- Telemetry (utility meter, smoke/burglar alarm, thermostat, babycam)

INTERNETWORKS

A collection of interconnected networks called internetwork or internet. A common form of internet is a collection of LANs connected by WANs. Subnet makes the most sense in the context of wide area network, where it refers to collection of routers to the communication lines owned by network operator. Telephone system consists of telephone switching offices connected to one another by high speed lines and houses and businesses by low speed lines. Lines and equipment owned by telephone companies form the subnet of telephone system. The combination of a subnet and its host forms a network. An internetwork is formed when distinct network are interconnected.

1.4 NETWORK SOFTWARE

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. Network software is now highly structured.

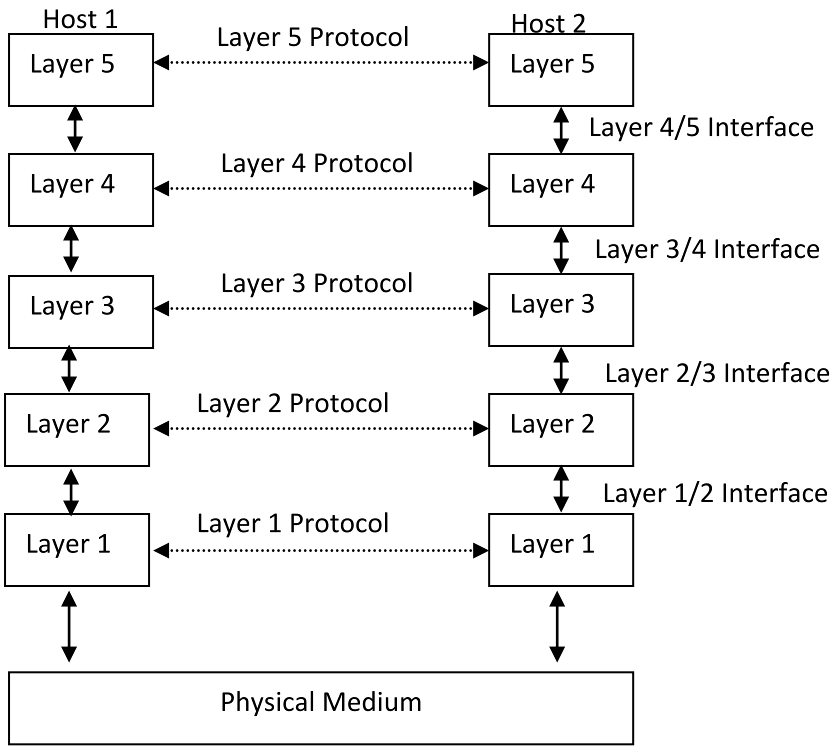
PROTOCOL HIERARCHIES

To reduce their design complexity, most networks are organized as a series of **layers or levels,** each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each 1ayer differ from network to network.

However, in all networks, the purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. Layer *n* on one machine carries on a conversation with layer *n* on another 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. Violating the protocol will make communication more difficult if not impossible. The entities comprising the corresponding layers on different machines are called **peers.** The peers communicate using protocol.

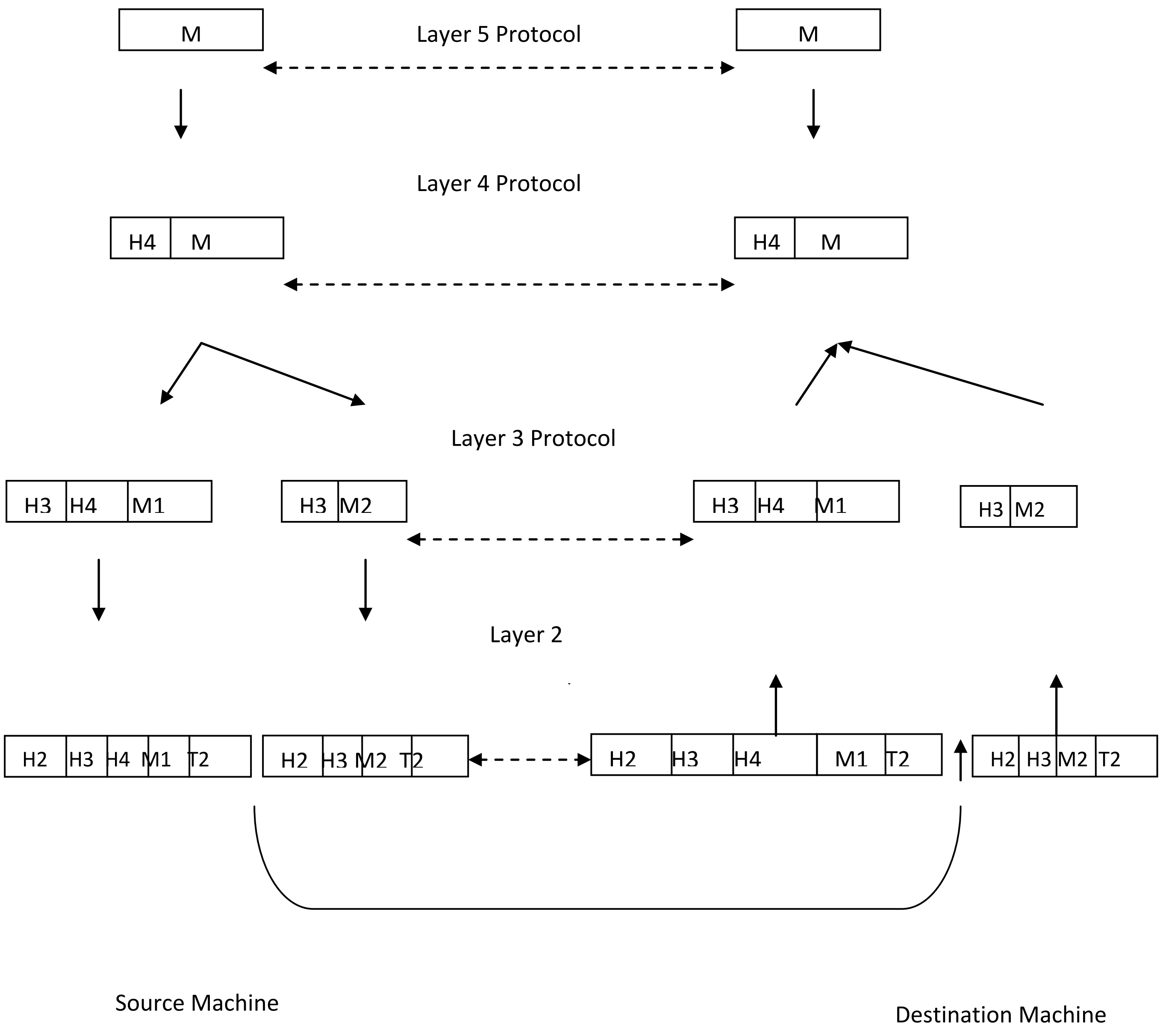
In reality, no data are directly transferred from layer n on one machine to layer *n on* another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached. Below layer 1 is the physical medium through which actual communication occurs. Between each pair of adjacent layers there is an **interface.** The interface defines which primitive operations and services the lower layer offers to the upper one. One of the most important considerations is defining clean interfaces between the layers. In addition to minimizing the amount of information that must be passed between layers, clean-cut interfaces also make it simpler to replace the implementation of one layer with a completely different implementation because all that is required of the new implementation is that it offers exactly the same set of services.



Layers, Protocols and Interfaces

A set of layers and protocols is called **network architecture**. The specification of architecture contains enough information to build the hardware/software for each layer so that it correctly obeys the appropriate protocol. A list of protocols used by a certain system, one protocol per layer, is called a protocol stack. A message M, produced by the application process puts a header in front of the message to identify the message and passes the result to the next layer. The header includes control information, such as a sequence numbers to allow the next layer in the destination machine to deliver messages in the right order. Headers may also contain sizes, times and other control fields. The layers break up the incoming messages into smaller units, packets.

For example, message *M* is split into two parts, m1 and m2. A Layer decides which of the outgoing lines to use and passes the packets to next layer. This Layer adds not only a header to each piece, but also a trailer, and give the resulting unit to layer below it for physical transmission. At the receiving machine the message moves upward, from layer to layer, with headers being stripped off as it progresses. None of the headers for layers below *n* are passed up to layer *n.* The peer process abstraction is crucial to all network design. Using it, the unmanageable task of designing the complete network can be broken into several smaller, manageable, design problems, namely the design of the individual layers.



DESIGN ISSUES FOR THE LAYERS

Some of the key design issues that occur in computer networking are present in several Layers. Every layer needs a mechanism for identifying senders and receivers. Since a network normally has many computers, some of which have multiple processes, a means is needed for a process on one machine to specify with whom it want to talk. As a consequence of having multiple destinations, some form of **addressing** is needed in order to specify a specific destination.

Another set of design decisions concerns the rules for **data transfer**. In some systems, data only travel in one direction **(simplex communication).** In others they can travel in either direction, but not simultaneously **(half-duplex communication).** In still others they travel in both directions at once **(full-duplex communication).** The protocol must also determine how many logical channels the connection corresponds to, and what their priorities are. Many networks provide at least two logical channels per connection, one for normal data and one for urgent data.

Error control is an important issue because physical communication circuits are not perfect. Many error-detecting and error-correcting codes are known, but both ends of the connection must agree on which one is being used. In addition the receiver must have some way of telling the sender which messages have been correctly received and which has not. Not all communication channels preserve the order of messages sent on them to deal with a possible loss of sequencing; the protocol must make explicit provision for the receiver to allow the pieces to be put back together properly. An issue that occurs at every level is how to keep a **fast sender from swamping a slow receiver with data**. Some of them involve some kind of feedback from the receiver to the sender, either directly or indirectly, about the receiver's current situation. This subject is called **flow control.**

Another problem that must be solved at several levels is the inability of all processes to accept arbitrarily long messages. This property leads to mechanisms for disassembling, transmitting, and then reassembling messages. A related issue is what to do when processes insist upon transmitting data in units that are so small that sending each one separately is inefficient. Here the solution is to gather together several small messages heading toward a common destination into a single large message and dismember the large message at the other side. When it is inconvenient or expensive to set up a separate connection for each pair of communicating processes, the underlying layer may decide to use the same connection for multiple, unrelated conversations. As long as this **multiplexing** and **de-multiplexing** is done transparently, it can be used by any layer. Multiplexing is needed in the physical layer, for example, where all the traffic for all connections has to be sent over at most a few physical circuits. When there are multiple paths between source and destination, a. route must be chosen. Sometimes this decision must be split over two or more layers.

CONNECTION-ORIENTED AND CONNECTIONLESS SERVICES

Connection-Oriented and Connectionless Services Layers can offer two different types of service to the layers above them:

1. Connection-oriented and

2. Connectionless.

1. CONNECTION-ORIENTED

This service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up. Similarly, to use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection. The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out in the same order at the other end.

2. CONNECTIONLESS SERVICE

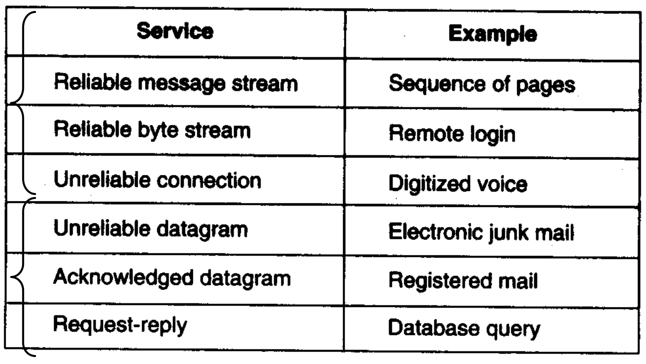
It is modeled after the postal system. Each message carries the full destination address, and each one is routed through the system independent of all the others. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first. Each service can be characterized by a **quality of service**. Some services are reliable in the sense that they never lose data. Usually, a reliable service is implemented by having the receiver acknowledge the receipt of each message, so the sender is sure that it arrived. The acknowledgement process introduces overhead and delays, which are often worth it but are sometimes undesirable.

A typical situation in which a reliable connection-oriented service is appropriate is file transfer. The owner of the file wants to be sure that all the bits arrive correctly and in the same order they were sent. Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former, the message boundaries are preserved when two 1-KB messages are sent, they arrive as two distinct 1-KB messages never as one 2-KB message. In the latter, the connection is simply a stream of bytes, with no message boundaries. Not all applications require connections.

Unreliable (not acknowledged) connectionless service is often called **datagram service,** which does not provide an acknowledgement back to the sender. In other situations, the convenience of not having to establish a connection to send one short message is desired, but reliability is essential. The **acknowledged datagram** service can be provided for these applications. Still another service is the **request-reply** service. In this service, the sender transmits a single datagram containing a request, the reply contains the answer. Request-reply is commonly used to implement communication in the client-server model: the client issues a request and the server responds to it.

Connection Oriented

Connectionless



Service	Example
Reliable message stream	Sequence of pages
Reliable byte stream	Remote login
Unreliable connection	Digitized voice
Unreliable datagram	Electronic junk mail
Acknowledged datagram	Registered mail
Request-reply	Database query

Six different types of Service

SERVICE PRIMITIVES

A service is formally specified by a set of **primitives** (operations) available to a user or other entity to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. One way to classify the service primitives is to divide them into four classes:

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

FIVE CLASSES OF SERVICE PRIMITIVES

First the server executes LISTEN to indicate that is prepared to accept the incoming connection. A common way to implement LISTEN is make it a blocking system call. After executing primitive a server process a block until a request for connection appears. A client process executed CONNECT to establish a connection with the server.

The CONNECT call needs to specify who to connect to, parameter gives the servers address. The operating system sends the packet to the peer asking it to connect as shown by (1) fig (refer class notes). The client process is connected until there is a response. When a packet arrives at the server it is processed by the operating system. When a system sees the packet is requesting a connection, it checks to see there is a listener. Unblock the listener and sends back the acknowledgement (2). The arrival of acknowledgement releases the client.

At this point the client and server are both are running and they have a connection established. Next step for the server to execute RECEIVE to prepare to accept the first request. The server does this immediately upon being released from the LISTEN, before the acknowledgement can get back to the client. The RECEIVE call blocks the server. The client executes send to transmit the request (3) followed by the execution to receive to get the reply.

The arrival of the request packet at the server machine unblocks the server process so it can process the request. After it has done the work it uses SEND to return the answer to the client (4). If it is done it use DISCONNECT to terminate the connection. An initial is a blocking call, suspending the client and sending a packet to the server saying that connection is no longer needed (5). When the server gets the packet it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection when the servers packet (6) gets back to the client machine, the client process is released and connection is broken.

THE RELATIONSHIP OF SERVICES TO PROTOCOLS

A ***service*** is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform or behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.

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

A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are implemented. A protocol relates to the *implementation* of the service and as such is not visible to the user of the service.

REFERENCE MODEL

2.1 INTRODUCTION

There are two Reference Models:

1. OSI Reference Model
2. TCP/IP Reference Model

2.2 THE OSI REFERENCE MODEL

2.2.1 THE PHYSICAL LAYER
2.2.2 THE DATA LINK LAYER
2.2.3 THE NETWORK LAYER
2.2.4 THE TRANSPORT LAYER
2.2.5 THE SESSION LAYER
2.2.6 THE PRESENTATION LAYER
2.2.7 THE APPLICATION LAYER

This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers The 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 . The OSI model has seven layers. The principles that were applied to arrive at the seven layers are as follows:

1. A layer should be created where a different level of abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally Standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.

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

2.2.1 THE PHYSICAL LAYER

The **physical layer** is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit, it is received by the other side as a 1 bit, not as a 0 bit.

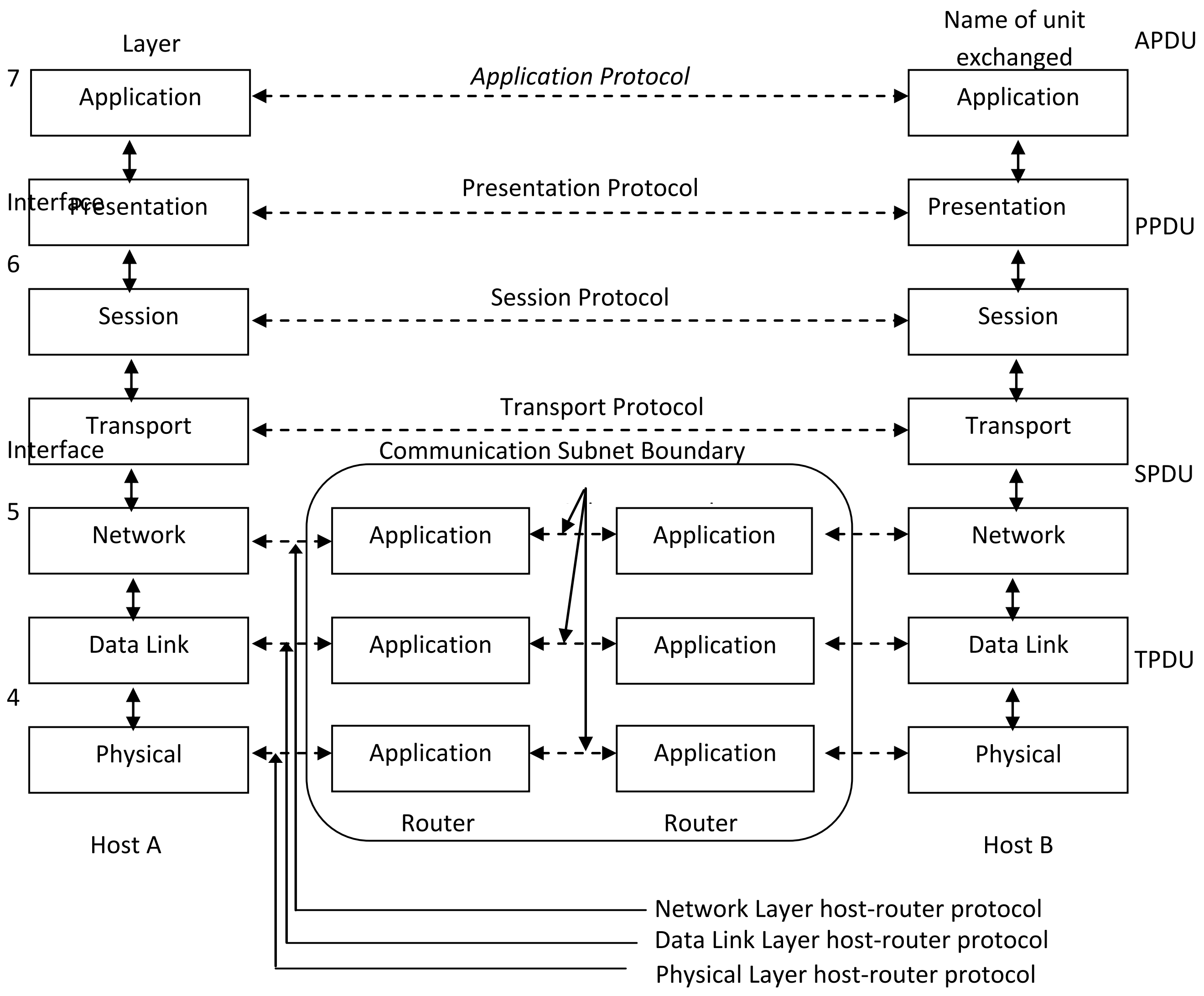
Typical questions here are how many volts should be used to represent a 1 and how many for a 0, how many microseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established and how it is torn down when both sides are finished, etc.,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

2.2.2 THE DATA LINK LAYER

The main task of the **data link layer** is to take a raw transmission facility and transform it into a line that appears free of undetected transmission errors to the network layer. It accomplishes this task by having the sender break the input data up into **data frames** transmit the frames sequentially, and process the **acknowledgement frames** sent back by the receiver.

Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning or structure, it is up to the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame.

If these bit patterns can accidentally occur in the data, special care must be taken to make sure these patterns are not incorrectly interpreted as frame delimiters. A noise burst on the line can destroy a frame completely. In this case, the data link layer software on the source machine can retransmit the frame.



OSI Reference Model

However, multiple transmissions of the same frame introduce the possibility of duplicate frames. A duplicate frame could be sent if the acknowledgement frame from the receiver back to the sender were lost. It is up to this layer to solve the problems caused by damaged, lost, and duplicate frames. The data link layer may offer several different service classes to the network layer, each of a different quality and with a different price.

Another issue that arises in the data link layer is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism must be employed to let the transmitter know how much buffer space the receiver has at the moment. Frequently, this flow regulation and the error handling are integrated. If the line can be used to transmit data in both directions, this introduces a new complication that the data link layer software must deal with. The problem is that the acknowledgement frames for *A* to *B* traffic compete for the use of the line with data frames for the B to A traffic.

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

2.2.3 THE NETWORK LAYER

The **network layer** is concerned with controlling the operation of the subnet. A key design issue is determining how packets are routed from source to destination. Routes can be based on static tables that are "wired into" the network and rarely changed. They can also be determined at the start of each conversation, for example a terminal session. Finally, they can be highly dynamic, being determined a new for each packet, to reflect the current network load. If too many packets are present in the subnet at the same time, they will get in each other's way forming bottlenecks. The control of such congestion also belongs to the network layer. There should be software that must count how many packets or characters or bits are sent by each customer. When a packet crosses between layers, with different rates on each side, the accounting can become complicated.

When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from the first one. The second one may not accept the packet because it is too large, the protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks get interconnected. In broadcast networks, the routing problem is simple, so the network layer often is thin or even nonexistent.

2.2.4 THE TRANSPORT LAYER

The basic function of the **transport layer** is to accept data from the session layer, split it up into smaller units if needed, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Under normal conditions, the transport layer creates a distinct network connection for each transport connection required by the session layer. If the transport connection requires a high throughput, however, the transport layer might create multiple network connections, dividing the data among the network connections to improve throughput. On the other hand, if creating or maintaining network connection is expensive, the transport layer might multiplex several transport connections onto the same network connection to reduce the cost. In cases, the transport layer is required to make the multiplexing transparent to the session layer. The transport layer also determines what type of service to provide the session layer and ultimately, the users of the network.

The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service are transport of isolated messages with no guarantee about the order of delivery, and broadcasting of messages to multiple destinations. The type of service is determined when the connection is established. The transport layer is a true end-to-end layer, from source to destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, the protocols are between each machine and its immediate neighbors, and not by the ultimate source and destination machines, which may be separated by many routers.

Many hosts are multiprogrammed, which implies that multiple connections will be entering and leaving each host. There needs to be some way to tell which message belongs to which connection. In addition to multiplexing several message streams onto one channel, the transport layer must take care of establishing and deleting connections across the network. This requires some kind of naming mechanism, so that a process on one machine has a way of describing with whom it wishes to converse. There must also be a mechanism to regulate the flow of information, so that a fast host cannot overrun a slow one. Such a mechanism is called **flow control** and plays a key role in the transport layer.

2.2.5 THE SESSION LAYER

The session layer allows users on different machines to establish **sessions** between them. A session allows ordinary data transport, as does the transport layer, but it also provides enhanced services useful in some applications. A session might be used to allow a user to log into a remote timesharing system or to transfer a File between two machines. One of the services of the session layer is to manage dialogue control. Sessions can allow traffic to go in both directions at the same time, or in only one direction at a time.

A related session service is **token management.** For some protocols, it is essential that both sides do not attempt the same operation at the same time. To manage these activities, the session layer provides tokens that can be exchanged. Only the side holding the token may perform the critical operation.

Another session service is **synchronization**; the session layer provides a way to insert checkpoints into the data stream, so that after a crash, only the data transferred after the last checkpoint have to be repeated.

2.2.6 THE PRESENTATION LAYER

The **presentation layer** performs certain functions that are requested sufficiently often to warrant finding a general solution for them, rather than letting each user solve the problems. The presentation layer is concerned with the syntax and semantics of the information transmitted. A typical example of a presentation service is encoding data in a standard way.

Most user programs do not exchange random binary bit strings. They exchange things such as people's names, dates, amounts of money, and invoices. These items are represented as character strings, integers, floating-point numbers, and data structures composed of several simpler items. Different computers have different codes for representing character strings (e.g., ASCII and Unicode), integers (e.g., one's complement and two's complement), and these different representations should be made to communicate. The presentation layer manages these abstract data structures and converts from the representation used inside the computer to the network standard representation and back.

2.2.7. THE APPLICATION LAYER

The **application layer** contains a variety of protocols that are commonly needed. For example, there are hundreds of incompatible terminal types in the world. Consider the plight of a full screen editor that is supposed to work over a network with many different terminal types, each with different screen layouts, escape sequences for inserting and deleting text, moving the cursor, etc. One way to solve this problem is to define an abstract **network virtual terminal** that editors and other programs can be written to deal with. To handle each terminal type, a piece of software must be written to map the functions of the network virtual terminal onto the real terminal.

For example, when the editor moves the virtual terminal's cursor to the upper left-hand corner of the screen, this software must issue the proper command sequence to the real terminal to get its cursor there too. All the virtual terminal software is in the application layer. Another application layer function is file transfer. Different file systems have different file naming conventions, different ways of representing text lines, and so on. Transferring a file between two different systems requires handling these and other incompatibilities. This work, too, belongs to the application layer, as do electronic mail, remote job entry, directory lookup, and various other general purpose and special-purpose facilities.

2.3 THE TCP/IP REFERENCE MODEL

ARPANET was a research network sponsored by the **DOD** (U.S Department of Defense), connecting hundreds of universities and government installations, using leased telephone lines. When satellite and radio networks were added later, the existing protocols had trouble interworking with them, so new reference architecture was needed. Thus, the ability to connect multiple networks in a seamless way was one of the major design goals from the very beginning. This architecture later became known as the TCP/IP Reference model. DOD wanted connections to remain intact as long as the source and destination machines were functioning.

2.3.1 THE INTERNET LAYER

All these requirements led to the choice of a packet-switching network based on a connectionless internetwork layer. This layer, called the internet layer, is the linchpin that holds the whole architecture together. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination. They arrive in a different order than they were sent, the job of higher layers to rearrange them. The analogy here is with the (snail) mail system. A person can drop a sequence of international letters into a mail box in one country and with a little luck, most of them will be delivered to the correct address in the destination country, letters will travel through one or more international mail gateways. Each country (i.e., each network) has its own stamps, preferred envelope sizes and delivery rules are hidden from the users. 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.

2.3.2 THE TRANSPORT LAYER

The layer above the internet layer in the TCP/IP model is the **transport layer**. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as OSI transport layer. Two end-to-end transport protocols are defined, 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 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 fast sender cannot swamp slow receiver with more messages than it can handle.

The second protocol is **UDP** (**USER DATAGRAM PROTOCOL)** is an un-reliable, connectionless protocol for applications protocols 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.

2.3.3 THE APPLICATION LAYER

On the top of the transport layer is 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 allows a user on one machine to log on to a distant machine and work there. The file transfer protocol provides the 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. Some other protocol are Domain Name System (DNS) for mapping host names on to their network address, NNTP, the protocol for moving USENET news article around, HTTP, the protocol for fetching pages on the World Wide Web.

2.3.4 THE HOST-TO-NETWORK LAYER

It tells only to point out that the host has to connect to the network using some protocol so it can send IP packets to it.

2.4 A COMPARISON OF THE OSI AND TCP/IP REFERENCE MODELS

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

Three concepts are central to the OSI Model:

1. Services

2. Interfaces

3. Protocols

The *services* 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 parameter are and what results to expect. Finally, the peer *protocols* used in a layer are the layer's own business. It can use any protocol it wants to, as long as it gets the job done. It can also change them at will without affecting software in higher layers.

The TCP/IP model did not distinguish between service, interface, and protocol. For example the only real services offered by the internet layer are SEND IP PACKET and RECEIVE IP PACKET.

- The OSI reference model was devised before the corresponding protocols were invented.
- With TCP/IP the reverse was true: the protocol came first, and the model was really just a description of the existing protocols.
- OSI model has seven layers and the TCP/IP has four layers. Both have (inter)network, transport, and application layers, but the other layers are different.
- OSI model supports both connection less and connection oriented communication in the network layer, but only connection oriented communication in the transport layer.
- The TCP/IP model has only one mode in the network layer but supports both modes in the transport layer giving the user a choice. This choice is especially important for simple request response protocols.

2.5 A CRITIQUE OF THE OSI MODEL AND PROTOCOLS

These lessons can be summarized as:

1. Bad timing
2. Bad technology
3. Bad implementations
4. Bad politics

1. Bad timing

The time at which a standard is established is absolutely critical to its success. When the subject is first discovered, there is a burst of research activity in the form of discussions, papers and meetings. After a while this activity subsides, corporation discover the subject, and the billion-dollar wave of investment hits. It is essential that the standard be return in the trough in between the two elephants. If the standards are written too early, before the research is finished, the subject may still be poorly understood; the result is bad standards. If they are written too late so many companies may have already made major investments in different ways of doing things that the standards are effectively ignored.

2. Bad technology

The choice of seven layers was more political than technical, and two of the layers (session and presentation) are nearly empty, whereas two other ones (data link and network) are overfull. The OSI model, along with the associated service definition and protocol, is extraordinarily complex. Another problem with OSI that some functions, such as addressing, flow control and error control, reappear again and again in each layer.

3. Bad Implementation

Complexity of the model and the protocols, it will come as no surprise the initial implementation were huge, unwieldy and slow.

4. Bad Politics

OSI was widely thought to be the creature of the European telecommunication ministries, the European community, and later the U.S. Government. Some people viewed this development in the same light as IBM announcing in the 1960's that PL/I was the language of the future or DoD correcting this later by announcing it was actually Ada.

2.6 A CRITIQUE OF THE TCP/IP REFERENCE MODEL

The TCP/IP model and protocols have their problems too.

- The model does not clearly distinguish the concepts of service, interface and protocol.
- The model is not all general and is poorly suited to describing any protocol stack other than TCP/IP.

- The Host-to-network layer is not really a layer at all in the normal sense of a term as used in the context of layered protocol. It is an interface between the network and the data link layer.
- TCP/IP model does not distinguish the physical and data link layer and they are completely different. The physical layer has to do with the transmission characteristic of copper wire, Fiber optics, wireless communication. The data link layer job's is to delimit the start and end of frames and get them from one side to the other with the desired degree of reliability.

Application Layer
Transport Layer
Network Layer
Data link Layer
Physical Layer

The Hybrid reference model

<u>POINT TO REMEMBER</u>

- Networks cab be divided into LAN,MAN,WAN.
- LAN covers a building and operate at high speeds.
- MAN cover a city eg. Cable television system.
- WAN covers a country or continent.
- LAN and MANs are unswitched(do not have routers) but WANs are switched.
- Protocol is nothing but a rules by which processes communicate.
- Protocols are either connectionless or connection-oriented.
- The Internet are evolved from the ARPANET.

<u>EXPECTED QUESTIONS</u>

<u>PART A</u>

1. Define Computer Networks?
2. Give two uses of Computer Networks?
3. Define Protocol?
4. What do u mean by connection oriented services?

5. What do u mean by Connection-less Services?
6. Expand LAN?
7. State any two topologies?
8. Draw the structure of Star Topology?
9. Expand WAN?
10. Define Flow Control?
11. State various layers in OSI Reference Model?
12. Give the use of Network Layer?
13. Expand OSI?
14. Write any on critique of OSI Reference Model?
15. Expand FTP?

PART B

1. Explain Protocol Hierarchies?
2. Write note on Connection Oriented and Connectionless Services?
3. State difference between Computer Networks and Distributed Systems?
4. Explain Design issues of the Layers?
5. Describe WAN in detail?
6. Write note on LAN?
7. Explain any two layers of OSI Reference Model?
8. Write note on Physical Layer?
9. State difference between Network and Transport Layer?
10. Explain the critique of the TCP/IP reference model?

PART C

1. Explain briefly about Uses of Computer Networks?

2. Explain briefly about Network hardware?

3. Explain briefly about Network software?

4. Explain briefly about OSI Reference Model?

5.. Write short notes on TCP/IP Reference Model?

UNIT II: PHYSICAL LAYER - Guided Transmission Media: Magnetic Media – Twisted Pair – Coaxial Cable – Fiber Optics. Wireless Transmission: Electromagnetic Spectrum Radio Transmission – Microwave Transmission – Infrared and Millimeter Waves – Light Waves. Communication Satellites: Geostationary, Medium-Earth Orbit, Low Earth-orbit. Satellites – Satellites versus Fiber**.**

OBJECTIVES

To Learn about

- What is the different type of cables.
- How to transfer information through network
- Communication satellite
- How wireless transmission works.

1. THE PHYSICAL LAYER

The purpose of physical layer is to transport a raw bit stream from one machine to another. Various physical media can be used for the actual transmission. Each one has its own niche in terms of bandwidth, delay, cost, and ease of installation and maintenance.

1.1 GUIDED TRANSMISSION MEDIA

1.1.1Magnetic Media

1.1.2Twisted Pair

1.1.3 **Coaxial Cable**

1.1.4Fiber Optics

Guided media:

Media are grouped into guided media such as copper wire and fiber optics, and unguided media, such as radio and lasers through air.

1.1.1 MAGNETIC MEDIA

(e.g., recordable DVDs) physically transport the tape or disks to he destination machine, and read them back in again. Although this method is not as sophisticated as using a geosynchronous communication satellite, it is more cost effective, and useful when high bandwidth or cost per bit transported is the key factor. Though, the bandwidth characteristics of magnetic tape are excellent, the delay characteristics are poor. Transmission time is measured in minutes, hours or even days.

1.1.2 TWISTED PAIR

Although the band width characteristics of magnetic tape are excellent, the delay characteristics are poor. Many applications need an on-line connection. The most common medium for transmission is **twisted pair**.

Structure:

- A twisted pair consists of two insulated copper wires about 1mm thick.
- The wires are twisted together in a helical form, the structure of a DNA molecule to avoid electrical interference to similar pairs of wires close by.
- Twisting is done because two parallel wires constitute a fine antenna. When the wires are twisted, the waves from different twists cancel out, so the wires radiates less effectively.

Application:

- The most common application of twisted pair is the telephone system. Nearly all telephones are connected to the telephone company (telco) office by a twisted pair. Wires used can run for several kilometers without amplification, but for longer distances repeaters are used. When many twisted pairs run in parallel for a substantial distance, such as all the wires coming from an apartment building to the telephone company office, there are bundled together and encased in a protective sheath. The pairs in the bundles would interfere with each other if they are not twisted.
- Twisted pairs can be used for either analog or digital transmission.
- The bandwidth depends on the thickness of the wires and the distance traveled. Due to their adequate performance and low cost, twisted pairs are widely used.

Categories:

Twisted pair cabling comes in several varieties, two of which are important for computer networks.

- **Category 3** twisted pairs consist of two insulated wires gently twisted together. Four such pairs are grouped in a plastic sheath to protect the wires and keep them together, category 3 cable running from a central **wiring closet** on each floor into each office. This scheme allowed up to four regular telephones or two multiline telephones in each office to connect to the telephone company equipment in the wiring closet.
- The most advanced **category 5** twisted pairs were introduced. They are similar to category 3 pairs, but with more twists per centimeter, which results in less crosstalk and a better-quality signal over longer distances, making them more suitable for high-speed computer communication.
- Up-and-coming categories are 6 and 7, which are capable of handling signals with bandwidths of 250 MHz and 600 MHz, respectively. All of these wiring types are often referred as **UTP** (**Unshielded Twisted Pair**), to contrast them with the bulky, expensive, shielded twisted pair cables IBM introduced in the early 1980s.

1.1.3 COAXIAL CABLE

- Another common transmission medium is the **Coaxial cable**.
- It has better shielding than twisted pairs, so it can span longer distances at higher speeds.

Categories:

Two kinds of coaxial cable are widely used. :

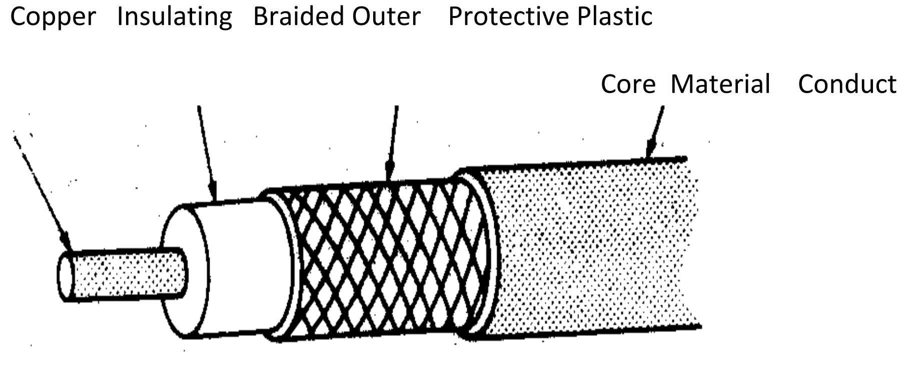
- One kind, 50-ohm cable, is commonly used when it is intended for digital transmission from the start.
- The other kind, 75-ohm cable, is commonly used for analog transmission and cable television but is
  becoming more important with the advent of Internet over cable.

Structure:

- A coaxial cable consists of a stiff copper wire as the core surrounded by an insulating material.
- The insulator is encased by a cylindrical conductor, often as a close woven braided mesh.
- The outer conductor is covered in a protective plastic sheath.
- The construction of the coaxial cable gives it a good combination of high bandwidth and
  excellent noise immunity.

The possible bandwidth depends on the cable quality, length and signal-to-noise ratio of the data signal.

Higher data rates are possible for shorter cables. Longer cables offer lower data rates. The coaxial cables are used within the telephone system for long-distance lines but have now largely been replaced by fiber optics on long- haul routes. Coax is still widely used for cable television and metropolitan area networks.



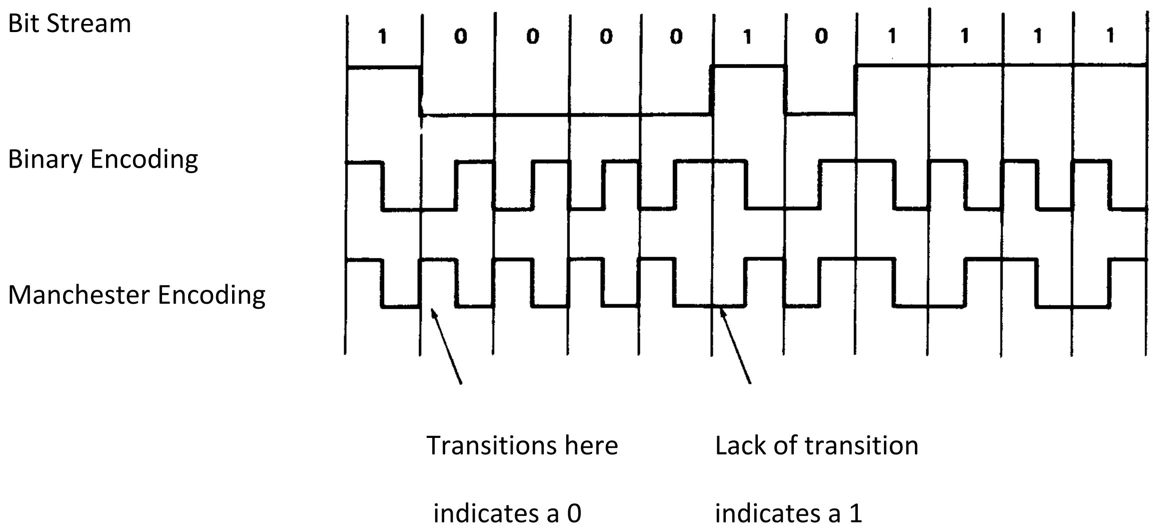
A Coaxial cable

Difference between Baseband & Broadband:

Baseband(digital)	Broadband(analog)
Baseband is simple and inexpensive to Install.	Broadband requires expensive radio Frequency engineers to plan the cable and Amplifier layout to install the system.
Maintenance cost is less.	Skilled personnel is required to maintain the System and periodically tune the amplifiers During its use.
It requires inexpensive interfaces.	The broadband interfaces are very expensive.
It offers a digital channel with a data rate Of about 10 Mbps over a distance of 1 km Using off-the-shelf coaxial cable.	Broadband offers multiple channels and can Transmit data, voice, so on in the same cable

Broadband Coaxial Cable:

The other kind of coaxial cable system uses analog transmission on standard cable television cabling. It is called **broadband.** Broadband refers to anything wider than 4kHz



Three different encoding systems

1.1.4 FIBER OPTICS

It has been made possible to transfer data by pulses of light. A Light signal is used to transmit a 1, the absence of light to transmit 0 bit. Visible light frequency is 10^8 MHz, so the bandwidth of an optical transmission is enormous.

There are three components in Optical transmission:

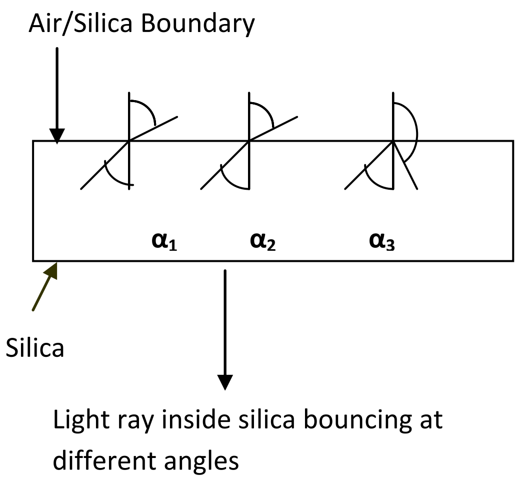
1. The transmission medium
2. The Light source
3. The detector

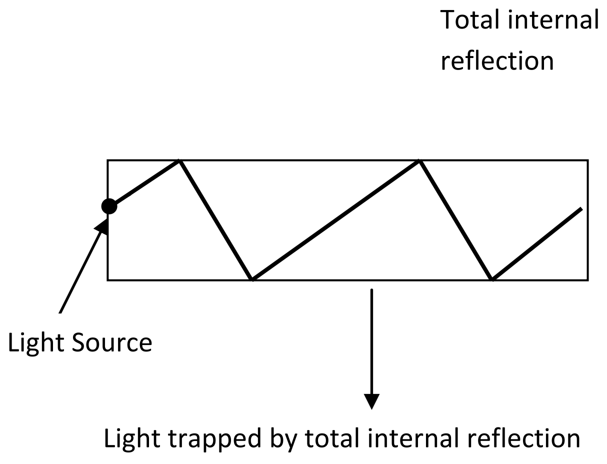
Process:

- The transmission medium is an ultra-thin fiber of glass or fused silica.
- The detector generates an electrical pulse when light falls on it.
- By attaching a light source to one end of an optical fiber and a detector to the other.
- User unidirectional data transmission system that accepts an electrical signal, converts and transmits
  it by light pulses, and then reconverts the output to an electrical signal at the receiving end.

- The transmission leaks light. When a light ray passes from one medium to another.

  For example, from fused silica to air, the ray is refracted (bent). The amount of refraction depends on the two medias. For angle of incidence above the certain critical value, the light is refracted back into the silica; none escapes into the air. Thus a light ray incident at or above the critical angle is trapped inside the fiber, and can propagate for many kilo meters with virtually no loss. Can be trapped inside without any loss.




Total internal reflection
Light Source
Light trapped by total internal reflection

Since any light ray incident on the boundary above the critical angle may be refracted internally, many different rays will be bouncing around at different angles. Each ray is said to have a different **mode** so a fiber having this property is called **Multimode fiber.** If the fiber's diameter is reduced to a few wavelength of light, the fiber acts like a wave guide, and the light will propagate in a straight line, without bouncing, yielding a **single mode fiber**. Single mode fibers are more expensive but are widely used for longer distances. Currently available single-mode fibers can transmit data at 50 Gbps for 100 km without amplification. Even higher data rates have been achieved in the laboratory for shorter distances.

<u>Transmission of Light through Fiber</u>

Optical fibers are of made of glass, which, in turn is made from sand, an inexpensive raw material available in unlimited amounts. Glassmaking was known to the ancient Egyptians, but their glass had to be no more than 1 mm thick or the light could not shine through. Glass transparent to be useful for windows was developed during the Renaissance. The glass used for modern optical fibers is so transparent that if the oceans were full of it instead of water, the seabed would be as visible from the surface as the ground is from an airplane on a clear day. The attenuation of light through glass depends on the wavelength of the light. For the kind of glass used in fibers, the attenuation in decibels per linear kilometer of fiber. The attenuation in decibels is given by the formula

$$\text{Attenuation in decibels} = 10 \log_{10} \frac{\text{Transmitted power}}{\text{Received power}}$$

For example, a factor of two loss gives an attenuation of $10 \log_{10} 2 = 3$ dB.

It shows the near infrared part of 0the spectrum, which is what is used in practice. Visible light has slightly shorter wavelengths, from 0.4 to 0.7 microns. The true metric purist would refer to these wavelengths as 400 nm to 700 nm, but we will stick with traditional usage. Three wavelength bands are used for optical communication. They are centered at 0.85, 1.3 and 1.55 microns respectively. The last two have good attenuation properties, the 085 micron band has higher attenuation, but at that wavelength the lasers and electronics can be made from the same material (gallium arsenide). All three bands are 25,000 to 30,000 GHz wide.

Light pulses sent down a fiber spread out in length as they propagate. This spreading is called chromatic dispersion. The amount of it is wavelength dependent. One way to keep these spread-out pulses from overlapping is to increase the distance them, but this can be done only by reducing the signaling rate. It has been discovered that by making the pulses in special shape related to the reciprocal of the hyperbolic cosine, nearly all the dispersion effects cancel out, and it is possible to send pulses for thousands of kilometers without appreciable shape distortion. These pulses are called **solitons**.

Fiber Cables

Fiber optic cables are similar to coax, except without the braid. At the center is the glass core through which the light propagates. In multimode fibers, the core is 50 microns in diameter, about the thickness of a human hair. In single-mode fibers, the core is 8 to 10 microns. The core is surrounded by a glass cladding with a lower index of refraction than the core, to keep all the light in the core. Next comes a thin plastic jacket to protect the cladding. Fibers are grouped in bundles, protected by an outer sheath.

Terrestrial fiber sheaths are normally laid in the ground within a meter of the surface, where they are occasionally subject to attacks by backhoes or gophers. Near the shore, transoceanic fiber sheaths are buried in trenches by a kind of seaplow. In deep water, they just lie on the bottom, where they can be snagged by fishing trawlers or attacked by giant squid.

Fibers can be connected in three different ways:

1. First, they can terminate in connectors and be plugged into fiber sockets. Connectors lose about 10 to 20 percent of the light, but they make is easy to reconfigure systems.
2. Second, they can be spliced mechanically. Mechanical splices just lay the two carefully-cut ends next to each other in s special sleeve and clamp them in place. Alignment can

be improved by passing light through the junction and then making small adjustments to maximize the signal. Mechanical splices take trained personnel about 5 minutes and result in a 10 percent light loss.

3. Third, two pieces of fiber can be fused (melted) to form a solid connection. A fusion splice is as good as a single drawn fiber, but a small amount of attenuation occurs. For all three kinds of splices, reflections can occur at the point of the splice, and the reflected energy can interfere with the signal.

Two kinds of light sources are used to do the signaling, LEDs (Light Emitting Diodes) and semiconductor lasers. They have different properties shown below.

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

They can be tuned in wavelength by inserting Fabry-perot or Mach-Zehnder interferometers between the source and the fiber. Fabry-perot interferometers are simple resonant cavities consisting of two parallel mirrors. The light is incident perpendicular to the mirrors. The length of the cavity selects out those wavelengths that fit inside an integral number of times. Mach- Zehnder interferometers separate the light in to two beams.

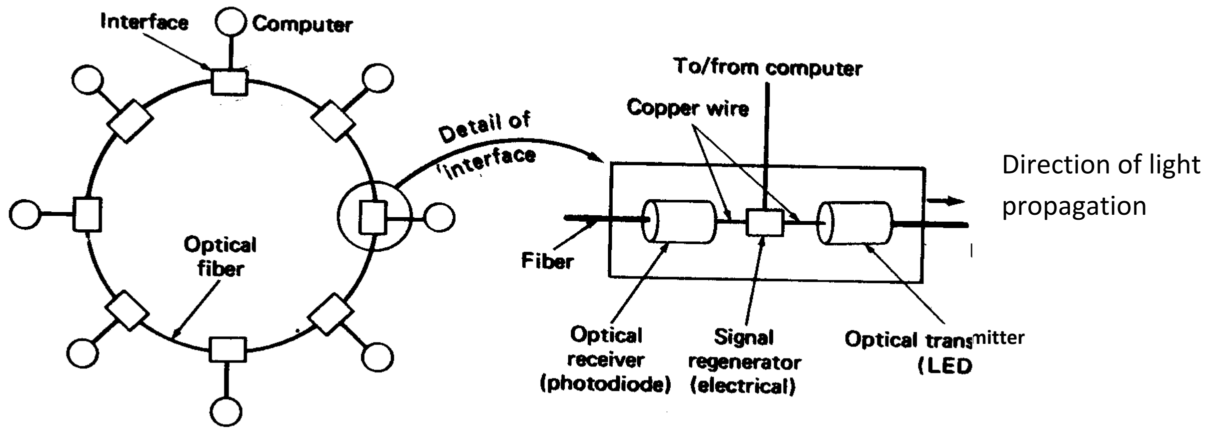
The two beams travel slightly different distances. They are recombined at the end and are in phase for only certain wavelengths. The receiving end of an optical fiber consists of a photodiode, which gives off an electrical pulse when struck by light. The response time of a photodiode is 1 nsec, which limits data rates to about 1 Gbps. Thermal noise is also an issue, so a pulse of light must carry energy to be detected. By making the pulse powerful, the error rate can be made arbitrarily small.

Fiber Optic Networks

Fiber optics can be used for LAN's but the technology becomes complex. The basic problem is that while vampire taps can be made on fiber LAN's by fusing the incoming fiber from the computer with the LAN fiber, the process of making a tap is very tricky and substantial light is lost. Another way is to treat as a ring network. The interface at each computer passes the light pulse stream through to the next link and also serves as a T junction to allow the computer to send and accept messages.

Two types of interfaces are used:

- A **passive interface** consists of two taps fused onto the main fiber. One tap has a LED or laser diode at the end of it for transmitting and the other has a photodiode for receiving. The tap itself is completely passive and thus extremely reliable because, a broken LED or photodiode does not break the ring, it just takes one computer off-line.
- The other interface type is the **active repeaters**. The incoming light is converted into an electrical signal, regenerated to full strength and retransmitted as light. The interface with the computer is an ordinary copper wire that comes with the signal regenerator. Purely optical repeaters are being used, these devices do not require the optical to electrical to optical conversions, which means they can operate at extremely high bandwidths. If an active repeater fails, the ring is broken down and the network goes down. On the other hand, the signal is regenerated at each interface, the individual computer-to-computer links can be kilometers long, with virtually no limit on the total size of the ring. The Passive interfaces lose light at each junction, so the number of computers and total ring length are greatly restricted.



A ring topology is not the only way to build a LAN using fiber optics. It is also possible to have hardware broadcasting using the **passive star** construction. In this design, each interface has a fiber running from its transmitter to a silica cylinder, with the incoming fibers fused to one end of the cylinder. Similarly, fibers fused to the other end of the cylinder are run to each of the receivers. Whenever an interface emits a light pulse, it is diffused inside the passive star to illuminate all the receivers, thus achieving broadcast.

In effect, the passive star performs a Boolean OR of all the incoming signals and transmits the result out on all lines. Since the incoming energy is divided among all the outgoing lines, the number of nodes in the network is limited by the sensitivity of the photodiodes.

Comparison of Fiber Optics and Copper Wire

FIBER	COPPER
It is thin and light weight, two fibers have more capacity and weight only 100 kg	One thousands twisted pairs 1 km long weight 8000 kg.
It can handle much higher bandwidth than copper	It can handle less bandwidth than fiber
Due to the low attenuation, repeaters are needed only about every 50 km on long lines.	Due to the low attenuation, repeaters are needed only about every 5 km for copper a substantial cost saving.
It is affected by corrosive chemicals in the air.	Copper is affected by power surges, electromagnetic interference or power failures.
It greatly reduces the need for expensive mechanical support systems, fiber wins hands down due to its lower installation cost. Fiber interfaces cost more than electrical interfaces.	It is inexpensive.

Fibers do not leak light and are quite difficult to tap. Fibers can be damaged easily by being bent too much.	The copper has excellent resale value to copper refiners who see it as very high grade ore.
Fiber excellent security against wire tappers, since optical transmission is inherently unidirectional, two-way communication requires either two fiber or two frequency bands on one fiber	Copper wire is used for two-way communications.

WIRELESS TRANSMISSION

2.1 The electromagnetic spectrum

2.2 Radio transmission

2.3 Microwave transmission

2.4 Infrared millimeter wavelength

2.5 Light wave transmission

1.2 wireless transmission

Mobile users , twisted pair, coax and fiber optics are no use

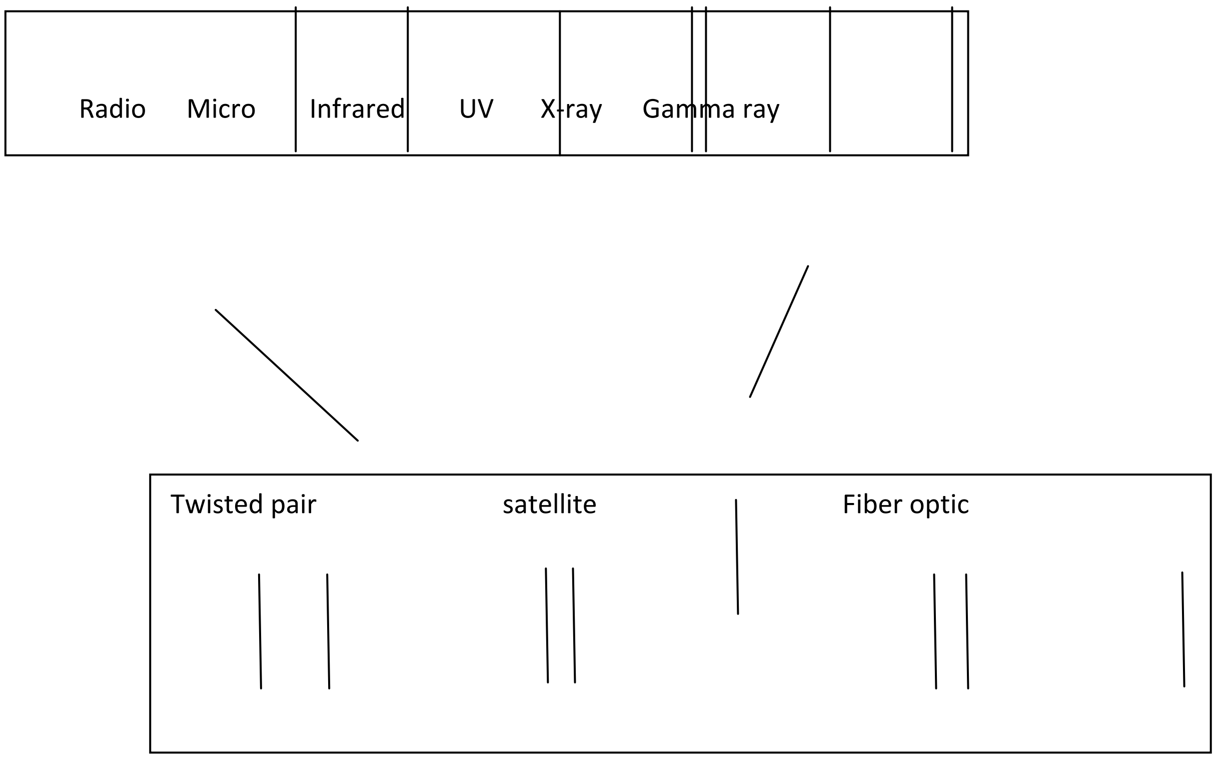
They need get the data for their laptop notebook, computer without being wired

- All mobile will use wireless
- Wireless may be better

2.1 The Electromagnetic spectrum

- When a electrons move they create electromagnetic waves that can propagate through space
- The no. of oscillations per second of a wave is called its **frequency 'f'**
- f is measured in HZ
- the distance between 2 consecutive maxima (or) minima is called the **wavelength**
- universally designed by Greek letter **λ**
- All electromagnetic waves travel at the same speed this speed is called the **speed of light**

F(Hz)



The radio, microwave, infrared, and visible light can all be used for transmitting information by modulating the amplitude frequency

X-ray and gamma dangerous to living things

LF-→ Band goes from 1 km to 10 km

LF ,Mf,HF-→ low medium and high frequency

2.2 <u>Radio transmission</u>

- Radio waves are easy to generate
- Can travel long distance
- And penetrate building easily
- Widely used for communication
- Radio wave also are Omni direction(travel in all direction)
- Some times this Omni direction is good
- Some times this Omni direction is bad
- The properties of radio waves pass through obstacles well

- High frequency radio waves travel in straight line
- Low frequency pass through obstacles well
- High frequency bounce off obstacles
- Due to radio's ability to travel long distances

2.3 Microwave transmission

- Microwave travel in a straight line if the tower are too far apart the earth will get in the way
- The higher the tower are apart they can be
- Unlike radio waves at lower frequency, microwave do not pass through buildings well
- Microwave communication is so widely used for long distance telephone communication
- Mobile phones television distribution and other uses that a severe shortage of spectrum has developed
- Microwave is also relatively inexpensive electromagnetic spectrum.
- ISM-→ Industrial, Scientific, Medical bands for unlicensed usage
- Cordless phones, radio- controlled toys, wireless mice and numerous other wireless households device use the ISM band.

2.4 Infrared and millimeter waves

- Unguided infrared and millimeter waves are widely used for short range communication
- The remote control used on televisions.
- VCRs, and stereos all use infrared communication
- Easy to build
- Major draw back: they do not pass solid objects
- Do not pass through solid walls
- You cannot control your neighbor's television with your remote control.
- No government license is needed to operate an infrared system.
- Infrared communication has a limited use on the desktop, for connecting note book computer and printer.

2.5 Light wave transmission

- ➢ Unguided optical signaling has been in use for centuries.
- ➢ The laser strength a very narrow beam
- ➢ Laser beam 1-mm wide at a larger the size of a pin head 500 meters
- ➢ **Dis adv:** laser beam cannot penetrates rain or thick fog, but they normally work well on suny days.

2. <u>COMMUNICATION SATELLITES</u>

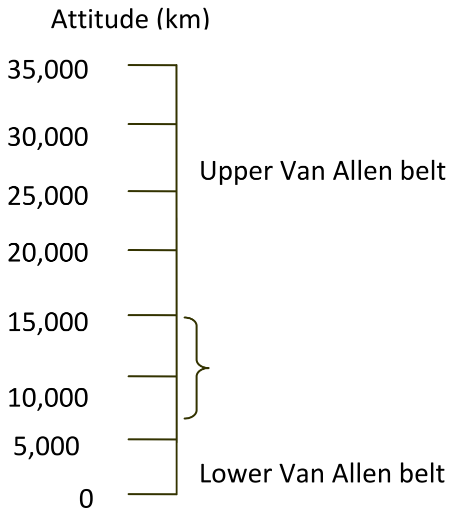
i. Geostationary satellites

ii. Medium-Earth Orbit satellites

3.1 Low-Earth Orbit Satellites

In the 1950s and early 1960s, people tried to set up communication systems by bouncing signals off metallized weather balloons. The received signals were too weak so the U.S navy noticed a kind of permanent weather balloon in the sky-the moon-and built an operation system for ship-to-shore communication by bouncing signals off it. The celestial communication field had to wait until the first communication satellite was launched. The key difference between an artificial satellite and a real one is that the artificial one can amplify the signals before sending them back, turning a strange curiosity in to powerful communication system.

Communication satellites and some of their properties, includes altitude above the earth, round-trip delay time, and number of satellites needed for global coverage.



Type	Latency (ms)	Sats needed
GEO	270	3
MEO	35-85	10
LEO	1-7	50

Communication satellites have some interesting properties that make them attractive for certain applications. A communication satellite can be thought of as a big microwave repeater in the sky. It contains several **transponders,** each of which listens to some portion of the spectrum, amplifies the incoming signal, and then rebroadcasts it at another frequency, to avoid interference with the incoming signal.

The downward beams can be broad, covering a substantial fraction of the earth's surface, or narrow, covering an area only hundreds of kilometers in diameter. This mode of operation is known as a bent pipe.

According to Kepler's law, the orbital period of a satellite varies as the radius of the orbit to the 3/2 power. The higher the satellite, the longer the period. Near the surface of the earth, the period is about 90 minutes. Low-orbit satellites pass out of view fairly quickly, so many of them are needed to provide continuous coverage.

A satellite's period is important, but it is not only issue in determining where to place it. Another issue is the presence of the Van Allen belts, layers of highly charged particles trapped by the earth's magnetic field. Any satellite flying with in them would be destroyed fairly quickly by the highly-energetic charged particles trapped there by the earth's magnetic field.

These factors lead to three regions in which satellite can be placed safely.

They are:

1. Geostationary satellites
2. Medium-Earth Orbit satellites
3. Low-Earth Orbit Satellites

iii. <u>GEOSTATIONARY SATELLITES</u>

- ✓ Geostationary Satellites including the orbits, solar panels, radio frequencies, and launch procedures. Unfortunately he concluded that satellites is impractical that due to the impossibility of putting power-hungry, fragile, vacuum tube amplifiers into orbit.
- ✓ The invention of the transistor changed all that and the first artificial communication satellite, Telstar, was launched in July 1962. Communication

satellites have become a multibillion dollar business and the only aspect of outer space that has become highly profitable.

- ✓ These high-flying satellites are called GEO (Geostationary Earth Orbit) satellites. With the current technology Geostationary satellites spaced much closer than 2 degrees in the 360-degree equatorial plane, to avoid interference.
- ✓ The effects of solar, lunar, and planetary gravity tend to move them away from their assigned slots and orientations, an effect countered by on-board rocket motors.
- ✓ This fine-tuning activity is called **station keeping**. When the fuel for the motors has been exhausted in about 10 years, the satellite drifts and tumbles helplessly, so it has to be turned off. The orbit decays and the satellite reenter the atmosphere and burns up or occasionally crashes to earth.
- ✓ Orbit slots are not only bone of contention. Frequencies are too, because the downlink transmissions interfere with existing microwave users.
- ✓ ITU has allocated certain frequency bands to satellite users. The C band was the first to be designated for commercial satellite traffic.
- ✓ Two frequency ranges are assigned in it, the lower one for **Downlink** traffic (from the satellite) and the upper one for **Uplink** traffic (to the satellite).
- ✓ To allow traffic to go both ways at the same time, two channels are required, one going each way.

Draw Dig( )

The principal satellite bands

Band	**Downlink**	**Uplink**	**Bandwidth**	**Problems**
L	1.5 GHz	1.6 GHz	15 MHz	Low bandwidth; crowded
Software	1.9 GHz	2.2 GHz	70 MHz	Low bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3500 MHz	Rain, equipment cost

a. The first geostationary satellites had a single spatial beam that illuminated about 1/3 of the earth's surface called its **footprint**. With the enormous decline in the price, size, and power requirements of microelectronics, a much more sophisticated broadcasting strategy has become possible. Each satellite is equipped with multiple antennas and multiple transponders.
b. Each downward beam can be focused on a small geographical area, so multiple upward and downward transmissions can take place simultaneously. These so-called **spot beams** are elliptically shaped, and can be as small as a few hundred km in diameter.
c. These **tiny** terminals have 1-meter or smaller antennas versus 10 m for a standard GEO antenna and can put out about 1 watt of power.
d. A special ground station, the **hub,** with a large high-gain antenna is needed to relay traffic between VSATs. In this mode of operation, either the sender or the receiver has a large antenna and a powerful amplifier. The trade-off is a longer delay in return for having cheaper end-user stations.
e.

iv. MEDIUM-EARTH ORBIT SATELLITES

At much lower altitudes, between the two Van Allen belts, the **MEO Satellites.** As viewed from the earth, these drift slowly in longitude, taking something like 6 hours to circle the earth. They must be tracked as they move through the sky. Because they are lowest than the GEOs they have a smaller footprint on the ground and require less powerful transmitters to reach them. The 24 **GPS** (**Global Positioning System**) satellites orbiting at about 18,000 km are examples of MEO satellites.

v. LOW-EARTH ORBIT SATELLITES

Moving down in altitude, we come to the **LEO** (**Low-Earth Orbit**) satellites. Due to their rapid motion, large numbers of them are needed for a complete system. On the other hand, because the satellites are so close to the earth, the ground stations do not need much power, and the round-trip delay is only a few milliseconds. Two ways, one is voice communication and other is at internet services.

Iridium:

- Iridium's business was providing worldwide telecommunication service using hand-held that communicate directly with the Iridium satellites.
- It provides voice, data, fax, and navigation service everywhere on land, sea, and air. Customers include the maritime, aviation, and oil exploration industries as well as people traveling in parts of the world lacking a telecommunications infrastructure.
- The Iridium satellites are positioned at an altitude of 750 km, in circular polar orbits.
- They are arranged in north-south necklaces, with one satellite every 32 degrees of latitude.
- With six satellite necklaces, the entire earth is covered.
- Each satellite has a maximum of 48 cells (spot beams), with a total of 1628 cells over the surface of the earth. Each satellite has a capacity 3840 channels, or 253,440 in all.

Globalstar:

- An alternative design to Iridium is Globalstar.
- It is based on 48 LEO satellites but uses a different switching scheme than that of Iridium.
- Whereas Iridium relays calls from satellite to satellite, which requires sophisticated switching equipment in the satellites, Globalstar uses a traditional bent-pipe design.
- The call originating at the North Pole is sent back to earth and picked up by the large ground station at Santa's workshop.
- This advantage of this scheme is that it puts much of the complexity on the ground, where it is easier to manage.
- The use of large ground station antennas that can put out a powerful signal and receive a weak one means that lower-powered telephones can be used. After all, the telephone puts out only a few milliwatts of power, so the signal that gets back to the ground station is fairly weak, even after having been amplified by the satellite.

Teledesic:

- **Teledesic,** is targeted at bandwidth-hungry Internet users all over the world.

- The goal of the Teledesic system is to provide millions of concurrent Internet users with an uplink of as much as 100 Mbps and a downlink of up to 720 Mbps using a small, fixed, VSAT-type antenna, completely bypassing the telephone system.
- The original design was for a system consisting of 288 small-footprint satellites arranged in 12 planes just below the lower Van Allen belt at an altitude of 1350 km.

Satellites versus Fiber:

A comparison between satellite communication and terrestrial communication is instructive. This glacial movement was caused in no small part by the regulatory environment in which the telephone companies were expected to provide good voice service at reasonable prices and in return got a guaranteed profit on their investment. Telephone companies began replacing their long-haul networks with fiber and introduced high-bandwidth services like ADSL (Asymmetric Digital Subscriber Line).

- First a single fiber has more potential bandwidth than all satellites ever launched, this bandwidth is not available to most users. The fibers that are now being installed are used within the telephone system to handle many long distance calls at once, not to provide individual users with high bandwidth. With satellites, it is practical for a erect an antenna on the roof of the building and completely bypass the telephone system to get high bandwidth. Teledesic is based on this idea.
- A second niche is for mobile communication. Many people nowadays want to communicate while jogging, driving, sailing, and flying. Terrestrial fiber optic links are of on use to them, but satellite links potentially are. That a combination of cellular radio and fiber will do adequate job for most users.
- A third niche is for situations in which broadcasting is essential. A message sent by satellite can be received by thousands of ground station at once. For example, an organization transmitting a stream of stock, bond, or commodity prices to thousands of dealers might find a satellite system to be much cheaper than simulating broadcasting on the ground.

- A fourth niche is for communication in places with hostile (stranger, enemy) terrain or a poorly developed terrestrial infrastructure. Launching one satellite was cheaper than stringing thousands of undersea cables among the 13,677 islands in the archipelago (group of islands).
- A fifth niche market for satellites is to cover areas where obtaining the right of way for laying fiber is difficult or unduly expensive.
- Sixth, when rapid deployment is critical, as in military communication systems in time of war, satellites win easily.

POINTS TO REMEMBER

- Transmission media can be guided or unguided.
- Unguided media include radio,microwaves,infrared and lasers through the air.
- An up-and-coming transmission syatem is satellite communication,eapecially LEO systems.
- A key element in most wide area network is telephone system.
- The first generation was analog ,dominated by AMPS.
- The Second generation was digital with D-AMPS,GSM and CDMA.
- The third generation will be digital and based on broadband CDMA.

EXPECTED QUESTIONS

PART A

1. Give one use of Physical layer.
2. What are the types of twisted pair cable?
3. State various categories of Co-axial Cable?
4. What is the use of Communication Satellite?
5. Give the expansion of MEO.
6. Give the expansion of VSAT.
7. What do you mean be VSAT?
8. Give the expansion of GEO.
9. What is the altitude (km) for GEO?
10. Give the formula to calculate "Attenuation in decibels".

PART B

11. Explain Magnetic Media?
12. Write note on Twisted Pair Cable?
13. State difference between Baseband and Broadband?
14. Explain Low-Earth Orbit Satellites?
15. Write about Fiber Optics?
16. State difference between Fiber and Copper?

PART C

1. Describe Guided Transmission media?
2. Write about Geostationary Satellites?
3. Explain briefly about Communication Satellites?

UNIT III: DATA-LINK LAYER Error Detection and correction – Elementary Data-link Protocols –Sliding Window Protocols. **MEDIUM-ACCESS CONTROL SUB LAYER:** Multiple Access Protocols – Ethernet – Wireless LANs - Broadband Wireless – Bluetooth.

OBJECTIVES

To Learn about

- What is mean by Error detection and correction.
- Different kinds of protocols used in Data-Link –Layer
- What is mean by Wireless LAN
- How Wireless LAN Differs from LAN.
- Usage of Broadband Wireless and Blutooth.

DATA LINK LAYER

2. ERROR DETECTION AND CORRECTION

The telephone system has three parts: the switches, the interoffice trunks, and the local

loops. The first two are now almost entirely digital in most developed countries. The local loops are still analog twisted copper pairs

a. **Error-Correcting Codes**

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

The former strategy uses **error-correcting codes and the latter** uses **error-detecting** codes. The use of error correcting codes is often referred to as forward error correction.

Each of these techniques occupies a different ecological niche.

On channels that are highly reliable, such as fiber, it is a frame consists of **m** data (i.e. message) bits and **r** redundant or check bits. Let the total length be *n* (i.e., **n=m +r)**. An **n** bit unit containing data and check bits is often referred to as **n-bit** codeword_

Given any two code words, say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just exclusive OR the two code words and count the number of 1 bits in the result,
for example:

```
 10001001
 10110001
---------------
00111000
```

The number of bit positions in which two codewords differ is called the Hamming distance (Hamming, 1950). Its significance is that if two code words are a Hamming distance *d* apart, it will require *d* single-bit errors to convert one into the other.

Parity→Error checking system that ensures safe data transmission between one computer to one another.

Parity bit→Extra bit is included with a set of binary digit as a check to see that all binary bit is transferred correctly.

Hamming code→It is the system of receiving data transmission where enough error checking information included with the data so that receiving system can automatically correct single bit mistakes with enquiring from the sender.

The error-detecting and error-correcting properties of a code depend on its Hamming distance.

To detect **d** errors, you need a distance **d+1** code because with such a code there is no way that **d** single-bit errors can change a valid codeword into another valid codeword.

When the receiver sees an invalid codeword, it can tell that a transmission error has occurred. Similarly, to correct **d** errors, you need a distance **2d + 1** code because that way the legal code words are so far apart that even with **d** changes, the original codeword is still closer than any other codeword, so it can be uniquely determined.

As a simple example of an error-detecting code, consider a code in which a single **parity** bit is appended to the data. The parity bit is chosen so that the number of **1** bits in the codeword is even (or odd). For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100. With odd parity 1011010 becomes 10110101.

A code with a single parity bit has a distance 2, since any single-bit error produces a codeword with the wrong parity. It can be used to detect single errors.

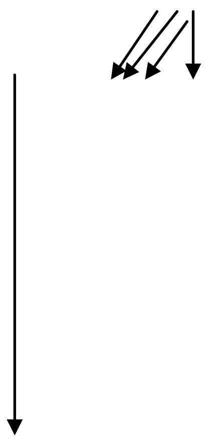
As a simple example of an error-correcting code, consider a code with only four valid codewords: 0000000000,0000011111,1111100000,and1111111111 This code has a distance **5**, which means that it can correct double errors. If the codeword 0000000111 arrives, the receiver knows that the original must have been 0000011111. If, a triple error changes 0000000000 into 0000000111, the error will not be corrected properly.

- Imagine that we want to design a code with **m** message bits
- and **r** check bits that will allow all single errors to be corrected.
- Each of the $\mathbf{2^m}$ legal messages has **n** illegal codewords at a distance 1 from it.
- These are formed by systematically inverting each of the **n** bits in the n-bit codeword formed from it.
- Thus, each of the $\mathbf{2^m}$ legal messages requires ***n*** + 1 bit patterns dedicated to it.
- Since the total number of bit patterns is $\mathbf{2^n}$, we must have $(n + l)2^m <= 2^n$. Using n =**m** + **r,** this requirement becomes $(m + r + 1) <= 2^r$ **.**Given **m,** this puts a lower limit on the number of check bits needed to correct single errors.

This theoretical lower limit can, in fact, be achieved using a method due to Hamming *(1950).* The bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on. The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits. The rest (3, *5, 6,* *7, 9,* **etc.)** are filled U with the ***m*** data bits. Each check bit forces the parity of some collection of bits, including itself, to be even (or odd). A bit may be included in several parity compJ1tatios. To see which check bits the data bit in position *k* contributes to, rewrite k as a sum of powers of 2.. For example, 11 = 1+2+8 and 29=1+4+8 + 16. A bit is checked by just those check bits occurring in its expansion (e.g., bit 11 is checked by bits 1, 2, and 8).

When a codeword arrives, the receiver initializes a counter to zero. It then examines each check bit, *k (k* = 1, 2, 4, 8, . .), to see if it has the correct parity - If the counter is zero after all the check bits have been examined (i.e., if they were all correct), the codeword is accepted as valid. If the counter is nonzero, it contains the number of the incorrect bit. For example, if check bits 1, 2, and 8 are in error, the inverted bit is 11, because it is the only one checked by bits 1, 2, and 8. Figure 3-7 shows some 7-bit ASCII characters encoded as 11-bit codewords using a Hamming code. Remember that the data are found in bit positions 3, *5,* 6, 7, 9, 10, and 11.

Char	**ASCII**	**Check bits**
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
C	1100011	11111000011
0	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101



Order of bit transmission

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

Hamming codes can only correct errors and correct burst errors

1. . A sequence of *k* Consecutive codeword's are arranged as a matrix, one codeword per row. Normally, the data would be transmitted one codeword at a time, from left to right. To correct burst errors, the data should be transmitted one column at a time, starting with the leftmost column.
2. When all *k* bits have been sent, the second column is sent, and so on, as indicated in Fig. 3-7. When the frame arrives at the receiver, the matrix is reconstructed, one column at a time.
3. If a burst error of length *k* Occurs, at most 1 bit in each of the *k* codewords will have been affected, but the Hamming code can correct one error per codeword, so the entire block can be restored
4. This method uses k*r* check bits to make blocks of *km* data bits immune to a Single burst error of length *k* or less.

3. Error-Detecting Codes

- Error-correcting codes are widely used on wireless links, which are notoriously noisy and error prone when compared to copper wire or optical fibers.
- Without error-correcting codes, it would be hard to get anything through. However, over copper wire or fiber, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error.
- This method can detect a single burst of length *n*, since only 1 bit per column will be changed. A burst of length *n* + 1 will pass undetected; however, if the first bit is inverted, the last bit is inverted, and all the other bits are correct. (A burst error does not imply that

  scheme may sometimes be adequate, in practice: another method is in widespread use: the **polynomial code,** also known as a **CRC (Cyclic Redundancy Check**). Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A k-bit frame is regarded as the coefficient list for a polynomial with *k* terms, ranging from x power k-1 to x power 0. Such a polynomial is said to be of degree *k*- 1. The high-order (leftmost) bit is the coefficient *of, x power k-1;* the next bit is the coefficient of *x power k-2,* and so on. For example, 110001 has 6 bits and thus represents a six-term polynomial_with coefficients 1, 1,0,0,0, and 1: x power 5 +x power 4 +x power 0. Polynomial arithmetic is done modulo 2, according to the rules of algebraic ory. There are no carries for addition or borrows for subtraction. Both addition and subtraction are identical to exclusive OR. For example:

10011011	00110011	11110000	01010101
+ll00l0l0	+11001101	—10100110	—10101111
01010001	11111110	01010110	11111010

Long division is carried out the same way as it is in binary except that the subtraction is done modulo 2, as above. A divisor is said “to go into” a dividend if the dividend has as many bits as the divisor. When the polynomial code method is employed, the sender and receiver must agree upon generator polynomial G(x), in advance. Both the high- and low- order bits of the generator must be I. To compute the **checksum** for some frame with *m* bits, corresponding to the polynomial *M(x),* the frame must be longer than the generator polynomial. The idea is to append checksum to the end of the frame in such a way that the polynomial represented by the check summed frame is divisible by G(x). When the receiver gets the check summed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error. The algorithm for computing the checksum is as follows:

1. Let *r* be the degree of G(x). Append *r* zero bits to the low-order end of the frame so it now contains *m* + *r* bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^r M(x)$, using modulo 2 division.
3. Subtract the remainder (which is always *r* or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial *T(x).*

Following Figure illustrates the calculation for a frame 1101011011 using the generator *G(x)x power 4 +x* **+** 1.It should be clear that *T(x)* is divisible (modulo 2) by G(x).

Frame : 1101011011

Generator: 1 0 0 11

Message after 4 zero bits are appended: 11010110110000

```
                         1100001010
1 0 0 1 1   1 1 0 1 0 1 1 0 1 1 0 0 0 0
          ___________________
         |    1 0 0 1 1
         |    -----------
               1 0 0 1 1
               1 0 0 1 1
               -----------
                0 0 0 0 1
                0 0 0 0 0
                -----------
                 0 0 0 1 0
                 0 0 0 0 0
                 -----------
                  0 0 1 0 1
                  0 0 0 0 0  ←
                  -----------
                   0 1 0 1 1
                   0 0 0 0 0
                   -----------
                    1 0 1 1 0
                    1 0 0 1 1
                    -----------
                         0 1 0 1 0
                      0 0 0 0 0
                      -----------
                       1 0 1 0 0
```

```
 1 0 0 1 1
   0 1 1 1 0
   0 0 0 0 0        Remainder
     1 1 1 0
```

Transmitted frame: 11010110111110

4. Elementary data link protocol:

- DL is concerned the packet passed across the interface to it from the network layer is pure data
- When the DL accepts a packet it encapsulates the packet in a frame
- Frame consists of an embedded packet, some control information (in the header) and a checksum the frame is then transmitted to the data link layer on the other machine.
- DL is waiting for something to happen by

Step 1: the procedure call wait-for event (& event). This procedure only returns when something has happened

Step 2: when a frame arrives at the receiver H/W computes the checksum. If the checksum is in correct there was a transmission error. the DL informed (event=checksum-error)Step 3: frame arrived un damaged, data L is also informed (event=frame-arrival)

Step 4: data structure are defined Boolean, sequence, packet, frame-kind, and frame.

Boolean -→ can take on the values true and false.

Sequence→ it is a small integer used to number the frames. These sequence number run from zero up to and including max-seq.

Packet: packet is the unit of information exchange between the network layer and the data Link layer on the same machine.

Frame: is composed of four fields.

- `Kind
- Sequence

- Acknowledgement (Ack)
- Information

The first three which contain control information and the last of which may contain actual data to be transferred. These control fields are collectively called the frame header.

```
# define MAX-PKT1024

Type def enum(false, true) Boolean;

Type def unsigned int seq_nr;

Type def struct { unsigned char data[MAXPKT]}

Type def enum {data,ack, nak} frame_kind;

Type def struct{

Frame_kind kind;

Seq_nr seq;

Seq_nr ack;

Pack info

} frame;
```

An unrestricted simplex protocol

- In this method data are transmitted in one direction only.
- Both the transmitting and receiving network layers are always ready.
- Processing time can be ignored.
- Infinite buffer space is available.
- The communication channel between the data link layer never damages or loses frames.
- 2 protocols consists of 2 distinct procedures a sender & a receiver.
- The sender runs in Dl layer of the source machine & the receiver runs in the DL layer of the destination machine.
- No acknowledgement is used here.

- ❖ The body of the loop consists of three action

1. Go fetch a packet from the network layer
2. Frame using variable S.
3. The info field of the frame is used by this protocol.

```
Type def enum {frame_arrival} event_type;

 # include “protocol.h”

Void sender 1(void)

{ frame S; packet buffer;

While (true) {

From_N/W _layer(Xbuffer); /go get something to send

S.info = buffer; → copy it into S for transmission

To_ physical_layer(&S);

} }

Void receiver 1(void)

{

Frame r;

Event_type_event (& event)

From_physical_layer(&r);

To_N/W_layer(&r.info);}}
```

A simplex stop and wait protocol

The ability of the receiving network layer to process incoming data infinitely quickly.

- ❖ The communication channel is still assumed to be error free.
- ❖ The main problem we have to deal with here is how to prevent this sender from flooding the receiver with data faster then the latter is able to process them

- ❖ The sender must never transmit a new frame until the old one has been fetched by from – physical layer. The new one over write the old one.
- ❖ Receiver provide feedback to the sender
- ❖ After having passed a packet to its network layer the receiver send a little dummy frame back to the sender which in effect gives the sender permission to transmit the next frame.
- ❖ A protocol in which the sender sends one frame and then waits for an ack. Before proceeding are called stop-&wait.

```
Void sender(void)

{frame S; packet buffer; even_type even;

While(true)

{

From_N/W_layer(&buffer);

S.info=buffer;to_physical_layer(&S);

Waitfor_event(&event);

}}

Void receiver (void)

{ frame r,S;event_type_event;

While (true){wait for_event(&event);

From physical_layer(&r);

To_N/W_layer(& r.info);

To_physical_layer(&S);}}
```

A simplex protocol for a noisy channel

- ❖ The sender could send a frame, but the receiver would only send an ack frame if the data were correctly receiver.
- ❖ If a damaged frame arrives at the receiver it would be discarded.

- This process would be repeated until the frame finally arrived intact .

Consider the following scenario

1. The network layer on A gives pack 1 to its data L layer. The packet is correctly received at B and passed to the N/W layer on B. B sends an ack frame back to A.
2. The ack frame gets lost completely. It just never arrives at all.
3. The DL layer on A eventually times out. Not having received an ack it assumes that its data frame was lost or damaged.
4. The duplicate frame also arrives at the data L layer B perfectly and is unwittingly passed to the N/W layer there. If A is sending a file to B, part of the file will be duplicated the protocol will fail.

Sliding window protocol

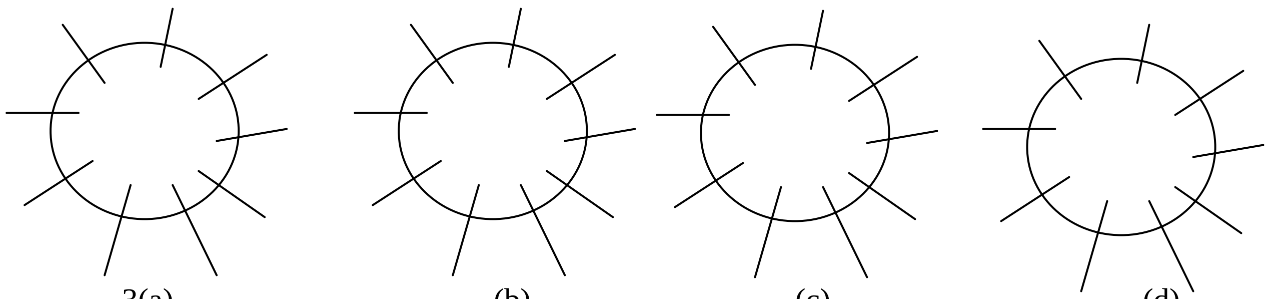
Transmitting data in both directions one way of achieving full-duplex data transmission is to have separate communication channel and use each one for simplex data traffic.

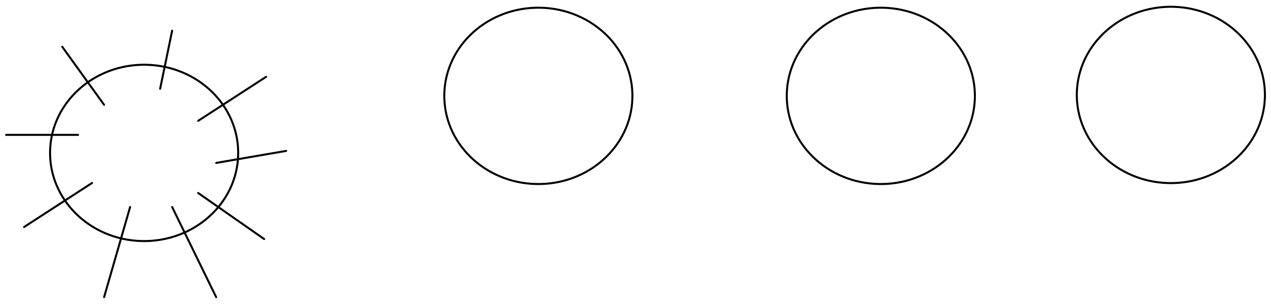
The data frames from A to B are intermixed, the ack frames from A to B header → incoming frame.The receiver → can tell whether the frame is data or ack.

The technique of temporarily delaying outgoing ack's, so that they can be hooked in to the next outgoing data frame is known as piggy backing.

The next three protocols are a bidirectional protocol that belongs to a class call sliding window protocols.

Sender maintains a set of sequence corresponding to frames. It is permitted to send these frames are said to be fall within the sending window receiver also maintain receiving window.


3(a) (b) (c) (d)



a) Initially

b) After the first frame has been send

c) After the first frame has been received

d) After the ack has been received

<u>Figure shows</u>

The sender must keep all these frames in its memory for possible retransmission.

The max window size is ‘n’ .the sender needs ‘n’ buffer to hold the ack frame.

Receiver when frame whose sequence number is equal to the lower edge of the window is received.

It is passed to the network layer an arc is generated and the window os rotated by one .the receiver window always remains at its initial size.

One-bit sliding window protocol

Next – frame –to-send tells which frame is the sender is trying to send.

Frame-expected → tells which frame the receiver is expecting

Physical-layer & start – timer procedure calls outside the main loop.

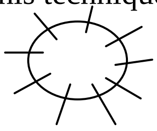
The arc field contains the number of the last received without error.

When the first valid frame arrives at computer B it will be accepted and frame expected will said to 1 all the frame will be rejected because B is now expecting frames with sequence no.=1

A protocol using go back N

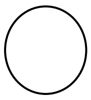
The product of these two factors basically tells what the capacity of the pipe is and the sender needs the ability to fill it without stopping in order to operate at peak efficiency .this technique is known as pipe lining.

Fig.(a)



- Frames 0 & 1 are collectively received an acknowledged
- Frame 2 however is damaged or lost
- The sender another of this problem continue to send frames until the timer for frame 2 expires
- The other general strategy for handling errors when frames are pipe lined is called selective repeat when it is used a bad frame that is received is discarded .but the good frame received after it are buffered
- When the sender times out only the older unack. frame is retransmitted
- If the frame arrives correctly the receiver can deliver to the network layer.

Fig (b)



- Frame 0 & 1 are again correctly received an acknowledged and frame 2 is lost
- When frame 3 arrives at the receiver the DL layer notices that is it has missed a frame so it sends back a NAK for 2 but buffer 3 when frames 4 & 5 arrive the NAK 2 gets back to the sender which immediately resends frame 2 .
- When that arrives the DL layer now has 2 3 4 5 can pass all of them to the N?W in the correct order.

A protocol using selective repeat

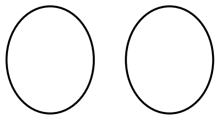
- ❖ This protocol handling errors is to allow the receiver to accept and buffer the frames following a damage or lost one
- ❖ In this protocol both sender and receiver maintain a window of acceptable seq.no.'s the senders window size starts out of a 0 and grows to some max seq.
- ❖ Arrival→telling whether the buffer is full or empty .whether a frame arrives its seq.no. is checked by the function between to see if it falls within the window .

a) initial situation with the window of size 7
b) after 7 frame have been send and received but not acknowledged
c) initial situation with a window of size 4

d) after 4 frames have been send and received but not acknowledged

the sender now transmits frames ‘0’ through ‘6’ .the receiver’s window allows it to accept any frame with seq.no. Between 0 & 6 inclusive. All 7 frames arrive correctly so the receiver acknowledges them and advance it window to allow receipt of 7 , 0 , 2 ,3 , 4 & 5.

The receiver send a piggy backed acknowledged for frame 6 .The sender is happy to learn that all its transmitted frame did actually arrive correctly. Frame 7 will be accepted by the receiver and its packet will be passed directly to the N/W layer.

Fig c & d 

Four bits are used for seq.no. This will range from 0 to 15 only. 8 acknowledged frames should be outstanding at any instant remaining frames are retransmission.

5. THE MEDIUM ACCESS CONTROL SUBLAYER:

❖ Broadcast channels are sometimes referred to as multi-access channels (or) random access channels.
❖ The protocols used to determine who goes next on a multi access channel belong to a sub layer of the DL layer called the MAC (medium access control)
❖ Which use a multi access channel as the basis for communication.

WAN→in contrast use point to point links

4.1 THE CHANNEL ALLOCATION PROBLEM.

Static Channel Allocation in LANs and Man’s

➢ The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is Frequency Division Multiplexing (FDM).
➢ If there are *N* users, the bandwidth is divided into *N* equal-sized portions (see Fig. 2-31), each user being assigned one portion.
➢ Since each user has a private *frequency* band, there is no interference between users. When there is only a small and constant number of users, each of which has a heavy (buffered) load of traffic (e.g., carriers’ switching

offices), FDM is a simple and efficient allocation mechanism. When the number of enders is large and continuously varying or the traffic is bursty, FDM presents some problems.

- If the spectrum is cut up into *N* regions and fewer than *N* users are currently interested in communicating, a large piece of valuable spectrum will be wasted.
- If more than *N* users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.
- Even assuming that the number of users could somehow be held constant at *N*, dividing the single available channel into static subchannel is inherently inefficient. The poor performance of static FDM can easily be seen from a simple queuing theory calculation. Let us Start with the mean time delay. ***T,*** for a channel of capacity ***C*** bps, with an arrival rate of frames/sec, each frame having a length drawn from an exponential probability density function with mean 1/µ frame/sec. With these parameters the arrival rate is λ frames/sec and the service rate is µ*C frames/sec.* From queuing theory it can be shown that for Poisson arrival and service times,
-

$$T = \frac{1}{\mu C - \lambda}$$

Now let us divide the single channel into *N* independent sub channels, each with capacity *C/N* bps. The mean input rate on each of the sub channels will now be

$$T_{FDM} = \frac{1}{\mu (C/N) - (\lambda / N)} = \frac{N}{(\mu C - \lambda)} = NT.$$

Dynamic Channel Allocation in LANs and MANs

It is worthwhile to carefully formulate the allocation problem. Underlying all the work done in this area are five key assumptions, described below. **Station Model**: The model consists of *N* independent stations (e.g., computers, telephones, or personal communicators), each with a program or user that generates frames for transmission. Stations are sometimes called **terminals**. The probability of a frame being generated in an interval of length Δt is $\lambda \Delta t$, *where* λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

<u>Single Channel Assumption</u>:- A single channel is available for all communication. All stations can transmit on it and all can receive from it. As far as the hardware is concerned, all stations are equivalent, although protocol software may assign priorities to them.

<u>Collision Assumption</u>:- If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a collision. All stations can detect collisions. A collided frame must be transmitted again later. There are no errors other than those generated by collisions.

<u>Continuous Time:-</u> Frame transmission can begin at any instant. There is no master clock dividing time into discrete intervals.

<u>Slotted Time</u>:- Time is divided into discrete intervals (slots). Frame transmissions always begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

<u>Carrier Sense</u>:- Stations can tell if the channel is in use before trying to use it. If the channel is sensed as busy, no station will attempt to use it until it goes idle.

<u>No Carrier Sense:-</u> Stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they deter mine whether the transmission was successful.

7. Multiple Access Protocols:

7.1 Carrier Sense Multiple Access Protocols

Persistent and Non persistent CSMA: - T

❖ **T**he 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 l-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 worse collision.
- The longer the propagation delay, the more important this effect becomes, and the the performance of the protocol. Even if the propagation delay is zero, there will still be collisions.
- If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends and then both will begin transmitting exactly simultaneously, resulting in a collision.
- If they were not so impatient, there would be fewer collisions. Even so, this protocol is far better than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame.
- This approach will lead to a higher performance than pure ALOHA. Exactly the same holds for slotted ALOHA.
- **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.
- 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.

7.2 CSMA with Collision Detection:

Persistent and non persistent CSMA protocols are clearly an improvement over ALOHA because they ensure that no station begins to transmit when it senses the channel busy. Another improvement is for stations to abort their transmissions as soon as they detect a collision. In other words, if two stations sense the channel to be idle and begin transmitting simultaneously, they will both detect the collision almost immediately. Rather than finish transmitting their frames, which are irretrievably garbled anyway, they should abruptly stop transmitting as soon as the collision is detected. Quickly terminating damaged frames saves time and bandwidth. This protocol, known as **CSMA/CD (CSMA with Collision Detection**) is widely used on LANs in the MAC sublayer. (Refer class notes for dig.) At the point marked t_0, 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. 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 Contention and transmission periods, with idle periods occurring when Stations are quiet (e.g., for lack of work). Suppose that two Stations both begin transmitting at exactly time t_0, the minimum time to detect the collision is then just the time it takes the signal to propagate from one station to another.

Consider the following worst-case scenario. Let the time for a signal to propagate between the two farthest stations be τ. At t_0, one station begins transmitting at t -ε, an instant before the signal arrives at the most distant station, that station also begins transmitting. 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 2τ-ε. In other words, in the worst case a station cannot be sure that it has seized the channel until it has transmitted for 2τ without hearing a collision. For this reason we will model the contention interval as a slotted ALOHA system with slot width 2τ. On a 1-km long coaxial cable, $\tau = 5\mu$sec. 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 2t sec. It is important to realize that collision detection is an *analog* process.

The station’s hardware must listen to the cable while it is transmitting. If what it reads back is different from what it is putting out, it knows that a collision is occurring. The implication is that the signal encoding must allow collisions to be detected (e.g., a collision of two 0-volt signals may well be impossible to detect). For this reason, special encoding is commonly used. It is also worth noting that a sending station must continually monitor the channel, listening for noise bursts that might indicate a collision. For this reason CSMA/CD with a single channel is inherently a half-duplex system. It is impossible for a station to transmit and receive frames at the same time because receiving logic is in use, looking for collisions during every transmission.

7.3 Collision Free Protocols

Collisions do not occur with CSMA/CD once a station has unambiguously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the cable is long (i.e., large τ) and the frames are short. And CSMA/CD is not universally applicable. In the protocols to be described, we assume that there are exactly N stations, each with a unique address from 0 to *N* -1 “wired” into it. It does not matter that some stations may be inactive part of the time.

7.4 A Bit-Map Protocol

In our first collision-free protocol, the **basic bit-map method**, each contention period consists of exactly *N* slots. If station 0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 during slot 1, but only if it has a frame queued. In general, station *j* may announce that it has a frame to send by inserting a 1 bit into slot *j*. After all *N* slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting in numerical order (refer class notes for Fig.)**.** Since everyone agrees on who goes next, there will never be any collisions.

After the last ready station has transmitted its frame, an event all stations can easily monitor another *N bit* contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols like this in which the desire to transmit is broadcast before the actual transmission are called **reservation protocols.** Let us briefly analyze the performance of this protocol. For convenience, we will measure time in units of the contention bit slot, with data frames consisting of *d* time units. Under conditions of low load, the bit map will simply be repeated over and over, for lack of data frames.

Consider the situation from the point of view of a low-numbered station, such as 0 or 1. When it becomes ready to send, the "current" slot will be somewhere in the middle of the bit map. On average, the station will have to wait N/2 slots for the current scan to finish and another full *N* slots for the following scan to run to completion before it may begin transmitting.

The prospects for high-numbered stations are brighter. Generally, these will

only have to wait half a scan (N/2 bit slots) before starting to transmit. High numbered stations rarely have to wait for the next scan. Since low-numbered stations must wait on average 1.*5N* slots and high-numbered stations must wait on average *0.5N* slots, the mean for all stations is *N* slots. The channel efficiency at low load is easy to compute. The overhead per frame is *N* bits, and the amount of data is *d* bits, for an efficiency of *d/ (N + d).* At high load, when all the stations have something to send all the time, the *N* bit contention period is prorated over *N* frames, yielding an overhead of only 1 bit per frame, or an efficiency of $d/(d + 1)$. The mean delay for a frame is equal to the sum of the time it queues inside its station, plus an additional $N (d + 1)/2$ once it gets to the head of its internal queue.

7.5 Binary Countdown:

A problem with the basic bit-map protocol is that the overhead is 1 bit per station, so it does not scale well to networks with thousands of stations. We can do better than that by using binary station addresses. A station wanting to use the channel now broadcasts its address as a binary bit string, starting with the high-order bit. All addresses are assumed to be the same length. The bits in each address position from different stations are BOOLEAN ORed together. We will call this protocol **binary countdown.** It was used in Datakit (Fraser, 1987). It implicitly assumes that the transmission delays are negligible so that all stations see asserted bits essentially instantaneously.

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

Fig. The binary countdown protocol. A dash indicates silence.

The channel efficiency of this method is *d/ (d* + log2 *N).* The frame format has been cleverly chosen so that the sender's address is the first field in the frame, even these $\log_2 N$ bits are not wasted, and the efficiency is 100 percent.

7.6 Limited-Contention Protocols

We have now considered two basic strategies for channel acquisition in a cable network: Contention, as in CSMA, and collision-free methods. Each strategy can be rated as to how well it does with respect to the two important performance measures, delay at low load and channel efficiency at high load. Under condition of light load, contention (i.e., pure or slotted ALOHA) is preferable due to its 1ow delay. As the load increases, contention becomes increasingly less attractive because the overhead associated with channel arbitration becomes greater. The reverse is true for the collision-free protocols. At low load, they have high delay, but as the load increases, the channel efficiency improves rather than gets worse as it does for contention protocols.

It would be nice if we could combine the best properties of the contention and collision-free protocols, arriving at a new protocol that used **Contention** at low load to provide low delay, but used a collision-free technique at high load to provide good channel efficiency. Such protocols, which we will call **limited-contention protocols,** Up to now the only contention protocols we have studied have been symmetric, that is, each station attempts to acquire the channel with some probability, p, with all stations using the same p. Interestingly enough, the overall system performance can sometimes be improved by using a protocol that assigns different probabilities to different stations.

Before looking at the asymmetric protocols, let us quickly review the performance of the symmetric case. Suppose that *k* stations are contending for channel access. Each has a probability p of transmitting during each slot. The probability that some station successfully acquires the channel during a given slot is then $kp(1-p)^{k-1}$. To find the optimal value of p, we differentiate with respect to p, set the result to zero, and solve for p. Doing so, we find that the best value of p is 1/k. Substituting p = 1/k, we get

Pr[success with $\left(\frac{\qquad}{k}\right)^{k-1}$ optimal p] = k-1

. For small numbers of stations, the chances of success are good, but-as soon as the number of stations reaches even five, the probability has dropped close to its asymptotic value of 1/e.

That the probability of some station acquiring the channel can be increased only by decreasing the amount of competition. They first divide the stations into (not necessarily disjoint) groups. Only the members, of group 0 are permitted to compete for slot 0. If one of them succeeds, it acquires the channel and transmits its frame. If the slot lies fallow or if there is a collision, the members of group 1 contend for slot 1, etc. By making an appropriate division of stations into groups, the amount of contention for each slot can be reduced, thus operating each slot near the left end. The trick is how to assign stations to slots. Before looking at the general case, let us consider some special cases. At one extreme, each group has but one member. Such an assignment guarantees that there will never be collisions because at most one station is contending for any given slot. We have seen such protocols before (e.g., binary countdown). The next special case is to assign two stations per group. The probability that both will try to transmit during a slot is p^2, which for small p is negligible. As more and more stations are assigned to the same slot, the probability of a collision grows, but the length of the bit-map scan needed to give everyone a chance shrinks. The limiting case is a single group containing all stations (i.e., slotted ALOHA). What we need is a way to assign stations to slots dynamically, with many stations per slot when the load is low and few (or even just one) station per slot when the load is high.

7.7 The Adaptive Tree Walk Protocol

One particularly simple way of performing the necessary assignment is to use the algorithm devised by the U.S. Army for testing soldiers for syphilis during World War II (Dorfman, 1943). In short, the Army took a blood sample from *N* Soldiers. A portion of each sample was poured into a single test tube. This mixed Sample was then tested for antibodies. If none were found, all the soldiers in the group were declared healthy. If antibodies were present, two new mixed samples ere Prepared, one from soldiers 1 through N/2 and one from the rest. The process was repeated recursively until the infected soldiers were determined.

For the Computerized version of this algorithm (Capetanakis, 1979), it is convenient to think of the stations as the leaves of a binary tree, as illustrated in slot 4-9. In the first contention slot following a successful frame transmission, slot 0, all stations are permitted to try to acquire the channel. If there is a collision, then during slot 1 only those stations falling under node 2 in the tree may compete.

If one of them acquires the channel, the slot following the frame is reserved for those stations under node 3. If, on the Other hand, two or more stations under node 2 want to transmit, there will be a collision during slot 1, in which case it is node 4’s turn during slot 2.

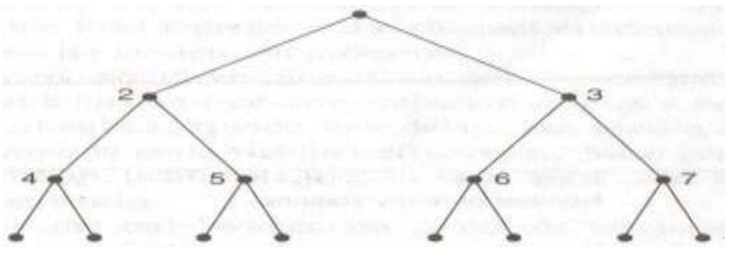


Figure - The tree for eight stations

If a collision occurs during slot 0, the entire tree is searched, depth first, to locate all ready stations.

Each bit slot is associated with some particular node in the tree. If a collision occurs, the search continues recursively with the node’s left and right children. If a bit slot is idle or if only one station transmits in it, the searching of its node can, stop because all ready stations have been located. To proceed, let us number the levels of the tree from the top, with node 1 in Fig. 4-9 at level 0, nodes 2 and 3 at level 1, etc. Notice that each node at level *i* has a fraction 2^{-i} of the stations below it. If the q ready stations are uniformly distributed, the expected number of them below a specific node at level *i* is 2^{-i}q. We would expect the optimal level to begin searching the tree as the one at which the mean number of contending stations per slot is 1, that is, the level at which $2^{-i} q = 1$. Solving this equation, we find that $i = \log_2 q$.

8. ETHERNET

The LAN market has seen several technologies but the most dominant today is Ethernet.

Three generations of Ethernet are available.

1. Traditional Ethernet (10 Mbps)
2. Fast Ethernet (100 Mbps)
3. Gigabit Ethernet (1000 Mbps)

We have seen one by one.

TRADITIONAL ETHERNET

Designed to operate at 10Mbps.Access to the network by a device is through a contention method (CSMA/CD).The media are shared all stations.

MAC Sub Layer:

The MAC sub layer governs the operation of the access method. It also frames data received form the upper layer and passes them to the PLS sub layer for encoding.

802.3 MAC Frame contains

1. Preamble
2. Start frame delimiter(SFD)
3. Destination address(DA)
4. Source Address(SA)
5. Length/type
6. Data
7. CRC

Preamble (7 bytes)

1. A frame contains 7 bytes (56 bits) of alternating 0's and 1's.
2. Receiving system to the coming frame and enable it to synchronize its input timing.
3. The pattern provides only an alert and a timing pulse.
4. The 56 bit pattern allows the stations to miss some bits at the beginning of the frame.
5. The preamble is actually added at the physical layer and is not part of frame.

Start Frame Delimiter (SFD)(1 Byte)

1. Signals the beginning of the frame.
2. The SFD tells the stations that they have a last chance for synchronization.

Destination Address (6 bytes)

Contains the physical address of the destination address.

Source Address (6 bytes)

Contains the physical address of the sender of the packet.

Length/Type

1. This field defined as a length or type field.
2. Value of field is less than 1518=> It is a length field and defines of the data field.
3. Value is greater than 1536=> The type of the PDU packet that is encapsulated in the frame.

DATA

Data Encapsulated from the upper layer protocols.

1. Minimum value = 46 bytes.
2. Maximum Value= 1500 bytes.

CRC

The last field contains error detection information

Frame Length

1. Minimum length restriction is required for the correct operations of CSMA/CD.
2. There is a collision, before the physical layer sends a frame out of station, it must be heard all the stations.
3. Collision is detected discard all the frames.
4. This situation frame length is diminishes in size since the smaller frames are

sent of faster.

5. The standard and smallest frame length for every 10Mbps Ethernet LAN is 512 bits or 64 bytes.
6. Standard and maximum length of frame is 1518 bytes.

Addressing

1. Each station has its own network interface card(NIC)
2. Represent as hexa decimal notation.

Ethernet address in hexa decimal notations.

06-01-02-01-2C-4B

Source address is always uni cast address. The frames comes from only one station.

Multicast or broad cast

The destination address,however can be uni cast,or broad cast.

Uni cast	---- one to one ---Relationship between sender and receiver.
Multicast	----one to many---- Defines a group of address.
Broad cast	---Special case of multi cast address.

The recipients are all the stations on the network.

Physical layer Implementation.

The destination address however can be unicast, multi cast or broad cast.

Common based implementation

1.10 Base 5

2.10 Base 2

3.10 Base-T

4.10 Base-FL

Now we have to see one by one.

10Base5: Thick Ethernet

1. First implementation is called 10Base5, thick Ethernet.
2. 10Base5 uses a bus topology with an external transceiver connected via a top to a thick coaxial cable. Maximum 500.

10Base2: Thin Ethernet

1. Second implementation is called 10 Base 2, thin Ethernet or cheaper net.
2. 10Base2 uses a bus topology with an internal transceiver or a point to point connection via an external transceiver.

3. Station uses internal transceiver means no need for an AUI cable.

10 Base-T Twisted pair Ethernet

1. Third implementation is called 10Base-T or twisted pair Ethernet.
2. Uses a physical star topology. The station are connected to a hub with an internal transceiver or an external transceiver.
3. Internal transceiver is used means no need for an AUI cable.
4. The interface card is directly connected to the medium connecter.
5. External transceiver is used, means transceiver is connected through an AUI cable. Then transceiver is connected to the hub.

10 Base-FL = Fiber link Ethernet

1. Several types of fiber optic 10 Mbps Ethernet are defined, the one implemented by vendors is called 10 BaseFL or Fiber link Ethernet.
2. Use star topology to connect stations to a hub.
3. Use External transceiver called fiber optic MAC. The station is connected to external transceiver by an AUI cable.
4. The transceiver is connected to the hub by using two pairs of fiber optic cables.

Bridged Ethernet

Ethernet evolution was the division of a LAN by bridges.Bridges have two effects on an Ethernet.They raise the bandwidth and separate the collision domain.

Raising the Bandwidth.

1. In an unbridged network,the total capacity(10 Mbps) is shared between all stations with a frame to send.
2. Only one station has frames to send,it benefits from the total capacity(10 Mbps)
3. More than one station needs to use means the capacity is shared.
4. The bridge divides the network into two or more networks.Bandwidth wise,each network is independently.
5. 12 stations are dividing into two networks. Each network has a capacity of 10 Mbps.

Switched Ethernet

The idea of a bridged LAN can be extended to a switched LAN. Instead of having two to four networks.

A switch Ethernet was a big step that opened the way to an even faster Ethernet.

Full Duplex Ethernet

The full duplex mode increase the capacity of each domain from 10 to 20Mbps.Two configuration modes are used one to transmit and one to receive.

No need for CSMA/CD

- No more need for carrier sensing.
- No more need for Collision Detection.

8.1 FAST ETHERNET

MAC SUBLAYER

The evaluation of Ethernet from 10 to 100Mbps.

Common Fast Ethernet implementations are

1.100 Base-x

(a)100Base-TX

(b)100Base-FX

2.100Base-T4

100Base-Tx

1. 100Base-TX uses Two pairs of twisted pair cable in a physical star topology.(UTP,STP)

(UTP-Un shielded twisted pair STP-Shielded Twisted Pair.)

The implementation allows an external transceiver or an internal transceiver.

TRANSCEIVER

It is responsible for transmitting,receiving,detecting,collisions and encoding/decoding of data.

ENCODING AND DECODIG

To achieve 100Mbps data rate,encoding and decoding is implemented.

- Encoder first perform block encoder.
- The data at the 125-mbps rate are then encoded into a signal using MLT====Multi transmission.

100 BASE –FX

Two pairs of fibre optic cable in a physical star topology.

TRANSCEIVER

It is responsible for transmitting,receiving,detecting,collisions and encoding/decoding of data.

Encoding and Decoding.

➢ The encoder first performs block encoding.

➢ The 4 parallel bits received from the NIC is encoded into 5 serial bits using 4B/5B.

➢ 125 MBPS rate are encoded into a signal using NRZ(Non Return Zero-Invert)

100 Base-T4

➢ Use of category 5 UTP or STP cable.

➢ New standard version use 3 or higher UTP.

Transceiver:

It is responsible for transmitting,receiving,detecting,collisions and encoding/decoding of data.

Encoding and Decoding.

Reduce the Band width.

Transmission using four wires.

➢ Two pairs are designed for unidirectional transmission and other two for bidirectional transmission.

➢ The two uni directional pairs are always free in one directional to carry collision signal.

8.3 GIGABIT ETHERNET

Higher data rate resulted in the design of the gigabit Ethernet protocol 1000Mbps.

MAC Sublayer

The whole idea in the evolution of Ethernet was to keep the MAC sublayer untouched.However,when it came to sending at a 1-Gbps rate,this was no longer possible.

Access Method.

Gigabit Ethernet has two distinctive approaches for medium access.Half duplex using CSMA/CD .Half duplex is complicated it is not used today.But in full duplex approach ,there is no need for CSMA/CD.All implementation of gigabit Ethernet follow the full duplex approach.

Physical layer implementation

Types are

Base-x

(a) 1000 Base-SX

(b) 1000 Base-LX

(c) 1000 Base-CX

2.1000-Base-T

Gigabit Ethernet can be categorized as either a two wire or a four wire implementation.

Two wire implementation is called 100Base-x.

- 1000-Base-SX= Used short wave optical fiber.
- 1000-Base-LX= Used Long Wave Optical Fiber.
- 1000-Base-CX= used short copper jumbers.

Four wire Implementation

1000Base-T = Twisted pair cable.

1000-Base-x

Both 1000Base-x and 1000Base-Lx use two fiber optic cables. The only difference between is that the former use short wave laser and the latter use long wave laser.

Transceiver

Functions are encoding,decoding,transmitting,receiving and collision detection.

Encoding

1.First performs block encoding.

2.Encoded into signal using NRZ encoding.

1000-Base-T

Use 5 UTP.

Four twisted pairs achieve a transmission rate of 1 Gbps.

Transceiver

- To send 1.25Gbps over four pairs of UTP 1000 Base-T uses an encoding scheme called 4D-PAM5(4 dimentional,5 level pulse amplitude modulation)
- The technical modulation is very difficult.

WIRELESS LAN

Wireless communication is one of the fastest growing technologies. The demand for connecting devices without cable is increasing everywhere. Wireless LAN is found on college campuses, office buildings, and public areas. At home, a wireless LAN can connect roaming devices to the Internet.

Two wireless technologies for LANs are

1.IEEE 802.11

Wireless LAN some times called Wireless Ethernet.

2.Bluetooth.

A complex technology for small wireless LAN.

Although both protocol needs several layers to operate,we mostly concentrate on the physical and data link layer.

1.1 IEEE 802.11

IEEE has defined the specification for a wireless LAN called 802.11

Physical Layer Specification.

In the physical layer we concentrate on

1.802.11 FHSS

2.802.11 DSSS

3.802.11a OFDM

4.802.11b HR-DSSS

5.802.11g OFDM

We have to one by one.

IEEE 802.11 FHSS

Frequency-hopping speed spectrum.

Sender sends on one carrier frequency for a short amount of time,then hops to another carrier frequency for the same amount of time,hops again to still another for the same amount of time and so on.

BAND

FHSS uses a 2.4 GHZ industrial, scientific and medical(ISM) band.

MODULATION AND DATE RATE

Modulation=1 Mbauds/s

Date Rate = 1 or 2 Mbps.

IEEE 802.11 DSSS.

Direct sequence spread spectrum(DSSS) method for signal generation in a 2.4GHZ ISM band.

To avoid buffering, however the time needed to send one chip code must be same as the time needed to send one original bit.

BAND

Use 2.4 GHZ ISM Band

Modulation and Data Rate:

Modulation==1 Mbaud/s technique use PSK.

Date Rate=1 or 2 Mbps.

IEEE 802.11a OFDM

It describes orthogonal frequency division multiplexing method for signal generation in a 5-GHZ ISM band.

OFDM

All the sub bands are used by one source at a given time.

BAND

- Use 5GHZ ISM band.
- Band is divided into 52 sub bands,with 48 sub bands for sending 48 groups of bits at a time,and 4 subbands for control information.
- If the sub bands are used randomly,security can also be increased.

MODULATION

OFDM uses psk and QAM for modulation.

DATA RATE

Date Rate are 18Mbps(psk) and 54Mbps(QAM).

IEEE 802.11b

It describes high rate DSSS(HR-DSSS) method for signal generation in a 2.4 GHZ ISM band.

HR-DSSS

It is similar to DSSS except for encoding method.Which is called complementary code keying(cck).CCK encodes 4 or 8 bits to one CCK symbol.

BAND

Uses 2.4 GHZ ISM band.

MODULATION

HR-DSSS defines four data rates.

(a)1 Mbps

(b) 2 Mbps

(c) 5.5 Mbps

(d)11 Mbps.

- The first two(a) and (b) use same modulation technique.
- The 5.5Mbops version uses BPSK and transmit a 1.375 Mbaud/s with 4 bit cck encoding.
- The 11Mbps version uses QPSK and transmit at 1.375 Mbaud/s with 8 bit cck encoding.

NOTE

The 11 Mbps version data rate close to 10 Mbps Ethernet.

IEEE 802.11g OFDM.

These is new specification uses OFDM with a 2.4GHZ ISM band.The complex modulation technique achieves a 54Mbps data rate.

MAC LAYER.

Use 2 MAC sub layers.

(1) Distributed coordination function.(DCF)

(2) Point coordination function(PCF)

FRAME TYPES

A wire less LAN defined by IEEE 802.11 has three categories of frames: management frames ,control frames, and data frames.

(1) MANAGEMENT FRAMES.

Management frames are used for initial communication between stations and access points.

(2) CONTROL FRAMES

Control frames are used for accessing the channel and acknowledging frames.

ADDRESSING MECHANISMS.

The IEEE 802.11 addressing is complicated.There are four cases defined by the value of two flags in the FC field,To DS and from DS.Each flag can be either 0 or 1,thus defining four different situations.The interpretation of the four address es(address 1 to address 4)in the MAC frame depends on the value of these flags.

Note that address 1 is always the address of the next device.Address 2 is always the address of the previous devices.Address 3 is the address of the final destination station if it is not defined by address 1.Address 4 is the address of the original source station if it is not the same as address 2.

CASE 1:

In this case,To DS=0 From DS=0.This means that the frame is not going to a distribution system(To DS=0) and is not coming form a distribution system(From DS=0).The frame is going from one station in a BS to another without passing through the distribution system.The ACK frame should be to the original sender.

CASE 2:

In this case,To DS=0 From DS=1.This means that the frame is Coming from a distribution system(From DS=1).The frame is coming from an AP and going to a station.The ACK should be sent to the AP..Note that address 3 contains the original sender of the frame.

CASE 3:

In this case, To DS=1 and from DS=0.This means that a frame is going to a distribution system(To DS=1).The frame is going from a station to an AP.The ACK is sent to the original station..Note that address 3 contains the final destination of the frame(in another BSS).

CASE 4:

In this case ,To DS=1 and From DS=1.This is the case in which the distribution system is also wireless.The frame is going from one AP to another AP in a wireless Distribution system.We don’t need to define address if the distribution system is a wired LAN because the frame in these cases has the format of a wired LAN frame(Ethernet,for example).Here,we need four addresses to define the original sender,the final destination,and two intermediate APs.

CSMA/CD

The CSMA/CD adds a procedure to handle a collision. In this method, any station can send a frame. The station then monitors the medium tom see if transmission was successful.

If any collision, the frame needs to be sent again. To reduce the probability of collision the second time, the station waits, it needs to back off.

It is reasonable that the station waits a little the first time, more if a collision occurs again, much more if it happens a third time and so on.

The back off method the station waits an amount of time between 0 and 2^N

* maximum propagation.

N=Number of attempted transmission.

The station that has a frame to send sets the back off parameter N to Zero. It sensese the line using one persistence strategies. After sending the frame, if it is does not hear any collision until the whole frame has been sent, the transmission is successful. Sends a jam signal to the line to inform other station of the situation to alert collision is occur.

All station discard the part of the frame received. station increments the value of the backoff parameter by 1.It checks to see if the value is exceeds the station abort the procedure.

Value is not exceed the limit, the station waits a random back off time based on the current value of the back off parameter and sense the line again.

CSMA/CA

The CSMA/CD avoids collision. The station uses one of the persistence strategies. After it finds the line idle, the station waits an(Interframe gap)amount of time. It then waits another random amount of time. After that , it sends the frame and sets a timer. The station waits for an acknowledgement from the receiver .If it receives the ack before the time expires, the transmission is successful. Station does not receive ack,it knows that some thing is wrong. The station increments the back off parameter, waits for a back off amount of time, and resends the line.

9. Broad band wireless

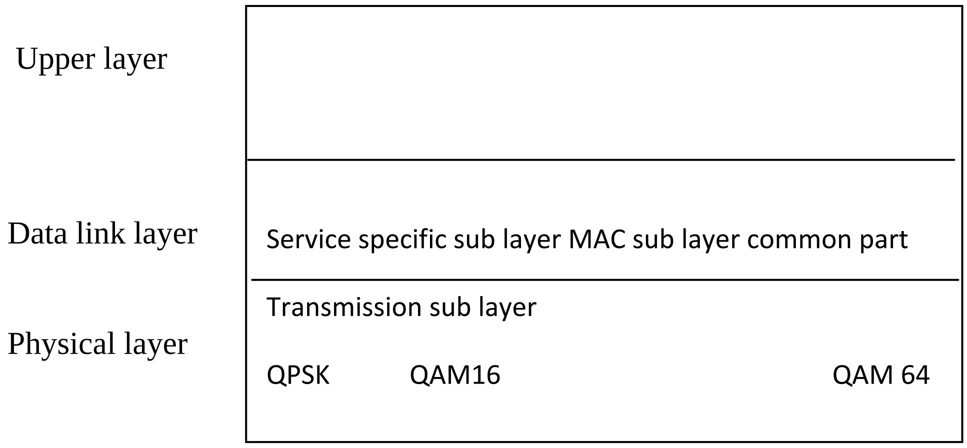
Electing a big antenna on a hill just outside of town and installing antennas directed at it on customers.

1. Comparison of 802.11 with 802.16
2. `802.16 PROTOCOL STACK
3. PHYSICAL LAYER
4. 802.16 mac SUBLAYER PROTOCOL
5. 802.16 Frame structure

1. Comparison of 802.11 with 802.16

802.16	802.11
Designed to provide high-bandwidth wireless communication	Designed to provide high-band with wireless commutation
Provide service to building	Deals with mobility radio wave
Can use full-duplex	Radio wave
Run over activity distance can be several kilometers	Communication over activity mean that security and privacy are essential
Business use	Support for real time traffic
Designed to be wireless cable television	Designed to be mobile ethernet

2. **The 802.16 protocol stack**



Data layer consist of 3 sub layers

- The bottom one deals with privacy and security
- It manages encryption, decryption and key management
- MAC sub layer main protocol
- Service specific -→ logical link sub layer in the other 802 protocols

3. **Physical layer**

QAM 64 → for close in subscribers

QAM 16→for medium distance subscriber

QPSK→for distance subscriber

4. **The 802.16 MAC sub layer protocol**

   Data link layer is divided into 3 sub layer

- Upper
- data link
- Physical

  1. Constant bit rate service
  2. Real time variable bit rate service
  3. Non-real time bit rate service
  4. Best effort service

1. Constant bit rate service

- Transmitting uncompressed voice
- This service needs to send a predetermined amount of data predetermined time intervals

2. Real time variable bit rate service

- Compressed multimedia and other real time applications
- Amount of band width is needed

3. Non-real time bit rate service

- For heavy transmission
- That not real time
- As large file transfers.

4. Best effort service

1. Constant bit rate service

- Transmitting uncompressed Voice
- This service needs to send a predetermined amount of data at predetermined time interval

2. Real time variable bit rate service

- Compressed multimedia and other real time applications
- Amount of bandwidth is needed

3. Non-real time bit rate service

- For heavy transmission
- That not real time

4. Best effort service

- Best effort subscribers
- If a request is success , it's will be noted
- If its not try again later
-

5. The 802.16 frame structures

   Start with enter-→ 1 bit

   Type →identify the frame types

   CT → indicate the presence (or) absence

   EK → tells which of the encryption is used

   Length → gives the complete length of the frame

   Connection → tells which connection this frame belong

   CRC → is a checksum

(a) 0 bit

Type is same * byte needed → allocated bytes for frames

10.BLUETOOTH:

Introduction of Bluetooth

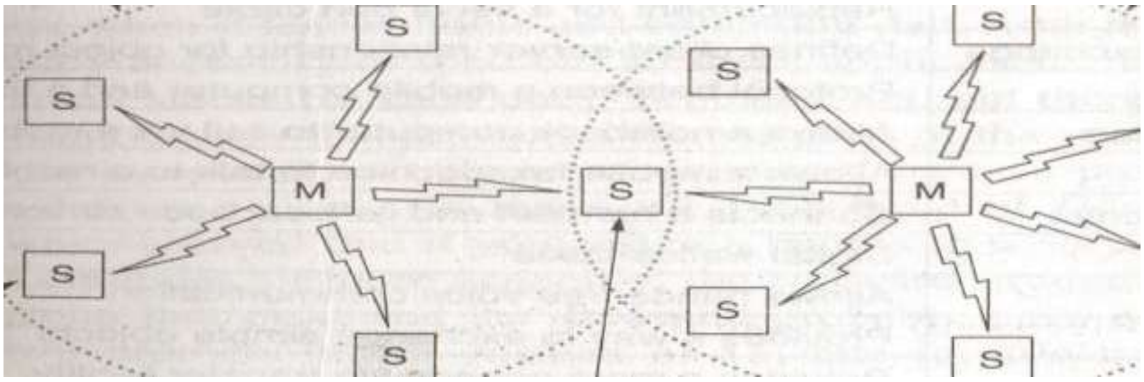
In 1994, the L. M. Ericsson company became interested in connecting its mobile phones to other devices (e.g., PDAs) without cables. Together with four other companies (IBM, Intel, Nokia, and Toshiba), it formed a SIG (Special Interest Group, i.e., consortium) to develop a wireless standard for interconnecting computing and communication devices and accessories using short-range, low-power, inexpensive wireless radios. The project was named **Bluetooth,** after Harald Blaatand (Bluetooth) II (940-981), a Viking king who unified (i.e., conquered) Denmark and Norway, also without cables.

By all this, in July 1999 the Bluetooth SIG issued a 1500-page specification of V1.0. Shortly thereafter, the IEEE standards group looking at wireless personal area networks, 802.15, adopted the Bluetooth document as a basis and began hacking on it.

The IEEE 802.15 committee is standardizing only the physical and data link layers: the rest of the protocol stack falls outside its charter. IEEE approved the first PAN standard, 802.15.1, in 2002, the Bluetooth SIG is still active busy with improvements. Although the Bluetooth SIG and IEEE versions are not identical, it is hoped that they will soon converge to a single standard.

1.2 Bluetooth Architecture

The basic unit of a Bluetooth system is a **piconet**, which consists of a master node and up to seven active slave nodes within a distance of 10 meters. Multiple piconets can exist in the same (large) room and can even be connected via a bridge node, as shown in Fig. 4-35. An interconnected collection of piconets is called a **scatternet**.



Bridge slave

Figure 4-35. Two piconets can be connected to form a scatternet.

The seven active slave nodes in a piconet, there can be up to 255 parked nodes in the net. These are devices that the master has switched to a low power state to reduce the drain on their batteries. In parked state, a device cannot do anything except respond to an activation or beacon signal from the master. There are also two intermediate power states, hold and sniff, but these will not concern us here. At its heart, a piconet is a centralized TDM system, with the master controlling the clock and determining which device gets to communicate which time slot. All communication is between the master and a slave, direct slave-slave communication is not possible.

1.3 Bluetooth Applications

Most network protocols just provide channels between communicating entities and let applications designers figure out what they want to use them for example, 802.11 does not specify whether users should use their notebook computers for reading e-mail, surfing the Web, or something else. In contrast, the Bluetooth V1.1 specification names 13 specific applications to be supported and provides different protocol stacks for each one. The 13 applications, which are called **profiles**, are listed in below

Name	Description
Generic access	Procedure for link management.
Service discovery	Protocol for discovering offered services.
Serial port	Replacement for a serial port cable.
Generic object exchange	Defines client-server relationship for object movement.
LAN access	Protocol between a mobile computer and a fixed LAN.
Dial-up networking	Allows a notebook computer to call via a mobile phones.
Fax	Allows a mobile fax machine to talk to a mobile phone.
Cordless telephony	Connects a handset and its local base station.
Intercom	Digital walkie-talkie.
Headset	Allows hand-free voice communication.
Object push	Provides a way to exchange simple objects.
File transfer	Provides a more general file transfer facility
Synchronization	Permits a PDA to synchronize with another computer.

Blue Tooth Layers

Bluetooth uses several layers that donot exactly match those of the internet model.

TDMA(Time division multiple access)

TDMA is a half duplex communication in which the slave and receive send and receive data, but don’t at the same time.

Example.

Walkie-takies using different carrier frequencies.

Single slave communication:

The piconet has only one slave,the TDMA operation is very simple. The time is divided into slots.

The master use even slots(0,2,4).

The slave use odd-numbered slots(1,3,5)

TDD-TDMA allows the master and the slave to communicate in half duplex mode.

In Slot 0 ==== Master sends and the slave receives.

In slot 1 ==== Slaves sends, the master receives. The cycle repeated.

Multiple slave communication.

More than one slave in the piconet.Again the master uses the even numbered slots,but a slave sends in the next odd numbered slot if the packet in the previous slot was addressed to it.

POINTS TO REMEMBER

- Data link layer convert the raw bit stream into frames.
- Bit –oriented protocols are SDLC,HDLC,ADCCP, or LAPB.
- PPP as the primary data link protocol over point-to-point lines.
- FDM –Dedicated a frequency band to each station.
- TDM –Deticated a time slot to each station
- CSMA/CA-Carrier sense multiple access with collision avoidance.
- A new development in LAN is VLAN

SUGGESTED QUESTIONS

Part-A

1. A second carrier sense protocol is------------- CSMA. In this protocol, a conscious attempt is made to be less greedy than in the previous one.

2. The-----------method each contention period consists of exactly *N* slots. If station 0 has a frame to send, it transmits a 1 bit during the zeroth slot.

3. The basic unit of a Bluetooth system is a -----------

4. ----------- codes are widely used on wireless links, which are notoriously noisy and error prone when compared to copper wire or optical fibers

5. Parity→Error checking system that ensures safe data transmission between one computer to one another.

6. Parity bit→extra bit is included with a set of binary digit as a check to see that all binary bit is transferred correctly.

7. What is Gateways?

Part-B

1. Define Framing?
2. Define Error control?
3. Define Flow control?

Part-C

1. Explain briefly about carrier-sense multiple access protocol and its types?
2. Explain briefly about collision-free protocols?
3. Explain briefly about Bluetooth architecture and applications?

UNIT IV: NETWORK LAYER: Routing algorithms – Congestion Control Algorithms. TRANSPORT LAYER: Elements of Transport Protocols – Internet Transport Protocols: TCP.

OBJECTIVES:

To learn about

- Network layer and its Components.
- Types of algorithms and control mechanism used to access the network layer
- Types of protocols and control mechanism used to access the Transport layer
- Transport layer and its elements.
- Types of Internet protocols includes UDP and TCP

1. THE NETWORK LAYER:

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, the network layer is the lowest layer that deals with end-to-end transmission.

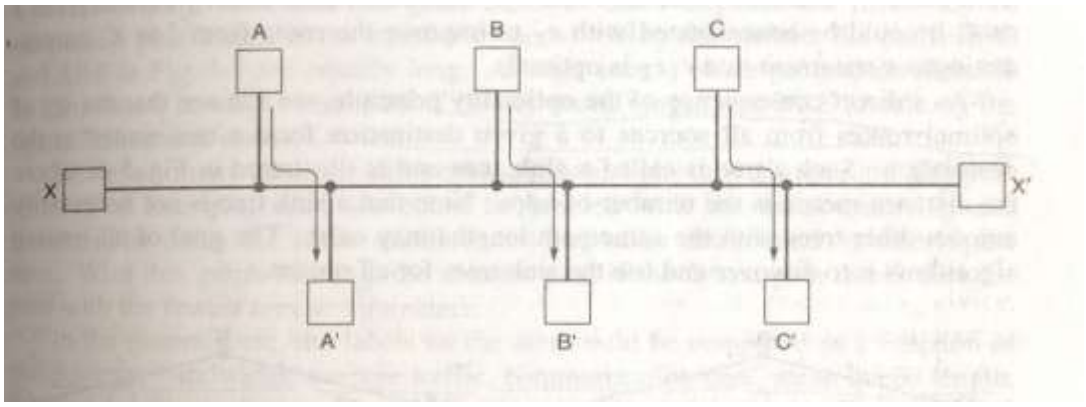
To achieve its goals, the network layer must know about the topology of the communication subnet (i.e., the set of all routers) and choose appropriate paths through it. It must also take care to choose routes to avoid overloading some of the communication lines and routers while leaving others idle.

2. ROUTING ALGORITHMS:

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. This is sometimes called **session routing** because a route remains in force for an entire user session (e.g., a login session at a terminal or a file transfer).

When each packet arrives, the process of looking up the outgoing line to use for fl in the routing tables. This process is forwarding. The other process is responsible for filling in and updating the routing tables. Certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and optimality. Correctness and simplicity hardly require comment, but the need robustness may be less obvious at first.

Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to equilibrium, no matter how long they run. A stable algorithm reaches equilibrium and stays there. Fairness and optimality may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals. In Fig. Suppose that there is enough traffic between *A* and *A’,* between *B* and *B’,* and between



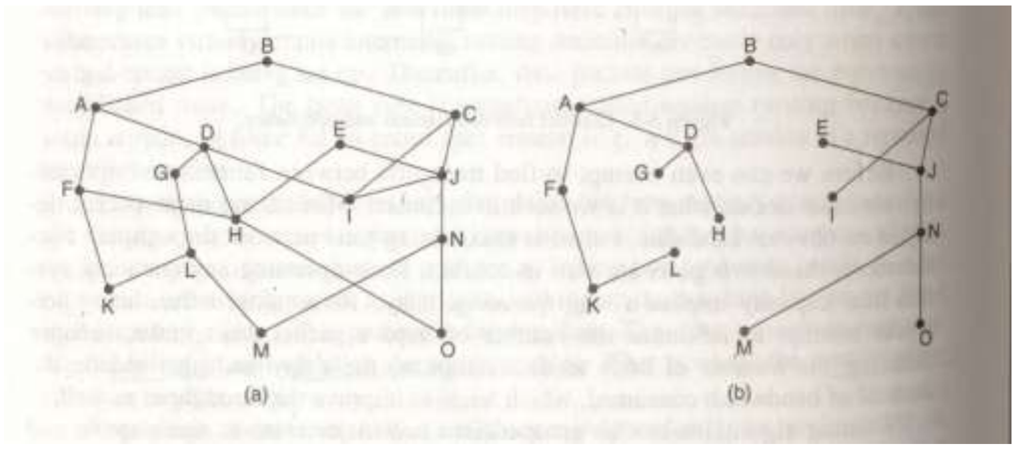
C and *C’* to saturate the horizontal links. To maximize the total flow, the X to X’ traffic should be shut off altogether. Unfortunately, X and x’ may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed.
Conflict between fairness and optimality.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive **Nonadaptive algorithms** do not base their routing decisThñ on niëas- urements or estiniates of the current traffic and topology. Instead, the choice of the toute to use to get from i to *J* (for all **I** and **.i)** is computed in advance, off-line, and downloaded to the routers when the ‘network is booted. This procedure is Sonletim called **static routing.**

Adaptive algorithms in contrast, change their routing decisions to reflect Changes in the topology, and usually the traffic as well. Adaptive 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., every *T* see, when the lo changes or when the topology changes), and what metric is used for (e.g., distance, number of hops, or estimated transit time).

2.1. <u>THE OPTIMALITY PRINCIPLE:</u>

One can make a general statement about optimal routes without regard to network or traffic. This statement is known as the **optimality principle**. 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 ***r1 r2*** is optimal. 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. (a)(b), where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. The goal of all routing algorithms is to discover and use the sink trees for all routers.



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

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. Links and routers can go down and come back up during operation, so different routers may have different ideas about the current topology.

2.2 <u>SHORTEST PATH ROUTING:</u>

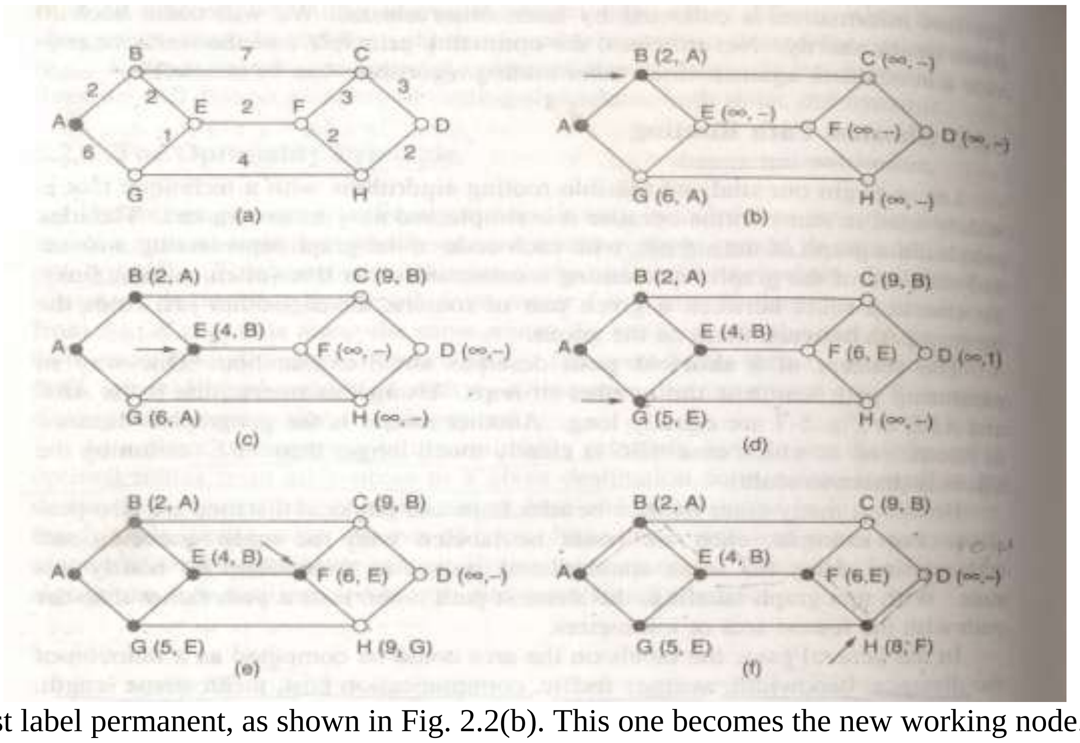
The concept of a shortest path deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths *ABC* and *ABE* in Fig. 5-7 are equally long. Another metric is the geographic distance in kilometers, in which case *ABC* is clearly much longer than *ABE* (assuming the figure is drawn to scale). Each arc could be labeled with the mean queuing and transmission delay. The labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the “shortest” path measured according to any one of a number of criteria or to a combination of criteria. There are several algorithms for computing the shortest path between two nodes of a graph. 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. Un- graph of Fig. *2-2(a)*, where the weights represent, for example, distance. We want to find the shortest path from *A* to

D. We start out by marking node *A* as permanent, indicated by a filled-in circle. Then we examine, in

turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to A.

Whenever a node is relabeled, we also label it with the node from which the each was made so that we can reconstruct the final path later. Having examined 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. 2.2(b). This one becomes the new working node.

2.2 The first five steps used in computing the shortest path from *A* to *D.* The arrows indicate the 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 be comes the working node for the next round. Figure 2.2 shows the first five steps of the algorithm.

To see why the algorithm works, look at Fig. 2.2(c). At that point we have just made ***E*** permanent. Suppose that there were a shorter path than ***ABE,*** Say *AXYZE.* 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. Either the label at Z is greater than or equal to that at ***E,*** in which case ***AXYZE*** cannot be a shorter path than ***ABE,*** or it is less than that of ***E,*** in which case ***Z*** and not ***E*** will become permanent first, allowing ***E*** to be probed from ***Z.***

2.2. <u>FLOODING:</u>

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

An alternative technique for damming the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. To achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

To prevent the list from growing without bound, each list should be augmented by a Counter, *k*, meaning that all sequence numbers through *k* have been seen. When a packet comes in, it is easy to check if the packet is a duplicate; if so it is discarded. Furthermore the full list below *k* is not needed, since ***k*** effectively summarizes it. **F**looding is not practical in most applications, but it does have some uses. For example, in military applications, distributed database applications, wireless networks.

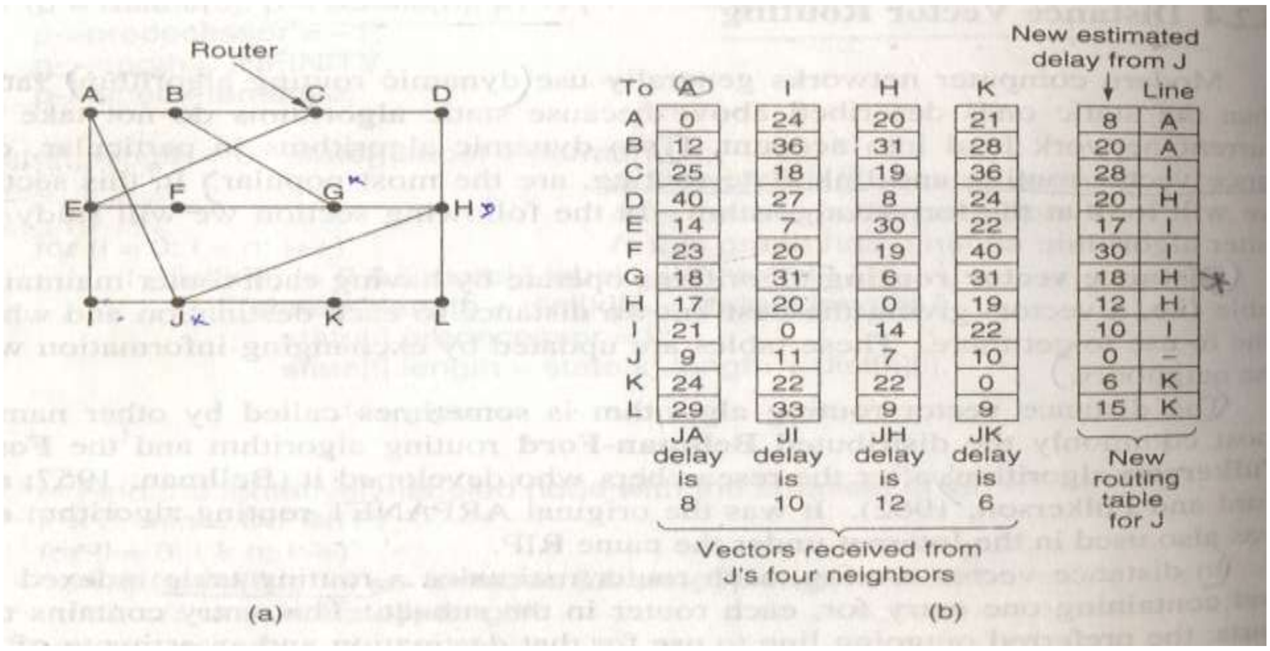
2.4 DISTANCE VECTOR ROUTING:

Modern computer networks generally use dynamic routing algorithm rather than the static c algorithms. Two dynamic algorithms in particular, distance vector routing and linear state routing, are the most popular. **Distance vector routing** algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to et there. These tables are updated by exchanging .information with the neighbors.

The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm and the **Ford Fulkerson** algorithm. In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet. This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or that destination.

An example, assume that is used as a metric and that the router the delay to each of its neighbors. Once every ***T*** msec each router sends to each neighbor a list of its estimated delays to each destination, it also receives similar list from each neighbor. Imagine that one of these tables has just come in from neighbor X, with X_i being X's estimate of how long it takes to get to route i. If the router knows that the delay to X is *m* msec, it also knows that it can reach router *i* via X in $X_i + m$ msec. By performing this calculation for each neighbor router can find out which estimate seems the best and use that estimate and a corresponding line in its new routing table.

This updating process is illustrated in Fig. 2.3. Part (a) shows a subnet The first four columns of part (b) show the delay vectors received from the neighb0 of router *J*. *A* claims to have a 12-nisec delay to *B*, a 25-msec delay to *C*, a 40. msec delay to *D*, etc. Suppose that *J* has measured or estimated its delay to its neighbors, *A*, *I*, *H*, and *K* as 8, 10, 12, and 6 msec, respectively.


Router
A
B
C
D
E
F
G
H
I
J
K
L
(a)
New estimated delay from J
Vectors received from J's four neighbors
(b)

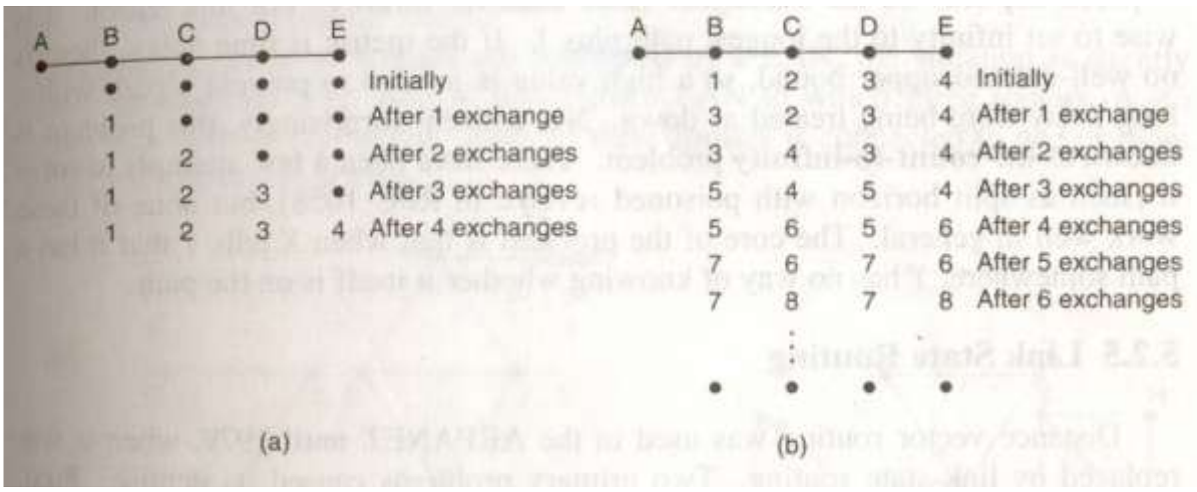
To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12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

2.3 (a) A subnet. (b) Input from *A, 1, H, K.* and the new routing table for *J.*

Consider how *J* computes its new route to router G. It knows that it can get to *A in 8* msec, and *A* claims to be able to get to G in 18 msec, so *J* knows it count on a delay of 26 msec 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) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H. The same calculation is performed for all the other destinations, with new routing table shown in the last column of the figure.

The count-to-infinity problem:

Distance vector routing works in theory but has a serious drawback in practice. Consider a router hose best route to destination X is large. If on the next change 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) subnet of Fig. 2.4, where the delay metric is the number of hops. Suppose *A* is down initially and all the other routers know this. In other words, they have all recorded the delay to *A* as infinity.


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)

2.4 The count-to-infinity problem.

When *A* comes up, the other routers learn about it via the vector exchanges. At the time of the first exchange, *B* learns that its left neighbor has zero delay to *A*. *B* flow makes an entry in its routing table 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. 2.4(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. Now let us consider the situation of Fig. *2.4(b)*, in which all the lines and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4 respectively Suddenly *A* goes down, or alternatively, the line between *A* and *B* is cut, Which is effectively the same thing from *B's* point of view. At the first packet exchange, *B* does not hear anything from *A*. Fortunately says: Do not worry; I have a path to *A* of length 2. Little does *B* know that path runs through *B* itself. For all *B* knows, *C* might have ten lines all with separate paths to *A* of length 2. As a result, *B* thinks it can reach *A* via *C*, with 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 the them at random and makes its new distance to *A* 4, as shown in the third row of Fig. 2.4(b). Subsequent exchanges produce the history shown in the rest of Fig.2.4(b).

2.5 LINK STATE ROUTING:

Distance vector routing was used in the ARPANET until 1979,when it was replaced by link state routing.there are two problems

1. The delya metric was queue length.
2. The algorithm often took to too long to converage

The idea behind link state routing is simple and can be stated as five parts.Each router muat do the following:

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

2.6 LEARNING ABOUT THE NEIGHBORS.

When a router is booted ,its first task is to learn who its neighbor are. It accomplishes this goal by sending a special HELLO packet on each point –to-point line.
The router on the other end is expected to send back a reply telling who it is .the names must be globally unique.

2.7 MEASURING LINE COST

The link state routing algorithm requires each router to know or at least have a reasonable estimate of the delay to each of its neighbors.The most direct way to determine this delay is to send over the line a special ECHO packet that the other is side is required to send back immediately

By measuring round trip timeand dividing it by two,

1. The sending router can get a reasonable estimateof the delay
2. Even better result the test can be conducted several times.

2.8 BUILDING LINK STATE PACKETS

Once the information needed for the exchange has been collected ,the next step is for each router to build a packet containing all the data.The packet starts with the identity of the sender,followed by a sequence number and age and a list of neighbours.
For each neighbor the delay to that neighbor is given

2.9 DISTRIBUTING THE LINK STATE PACKETS

The trickiest part of the algorithm is distributing the link state packet reliably.As the packets are distributed and installed the routers getting the first ones will change their routes

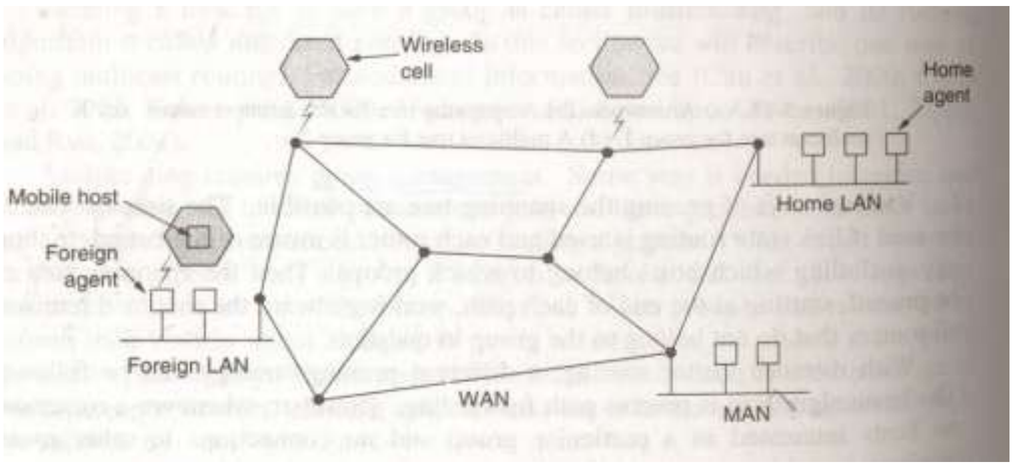
2.10 COMPUTING THE NEW ROUTERS

Once a router has accumulated a full set of link state packets,it can construct the entire subnet graph because every link is represented. Every link is,in fact represented twice,once for each direction. The two values can be averaged or used separately.

2.11 ROUTING FOR MOBILE HOSTS:

This section deals with the subject of incorporating mobile hosts into a network. The model of the world that network designers typically use is shown in Fig. 5-18. Here we have a WAN consisting of routers and hosts.

Hosts that never move are said to be stationary. They are connected to the network by copper wires or fiber optics. Migratory hosts are basically stationary hosts who move from one fixed site to another from time to time but use the network only when they are physically connected to it. Routing hosts actually compute on the run and want to maintain their connections as they move around. The **mobile hosts** mean either of the latter two categories, that is, all hosts that are away from home and
still want to be connected. All hosts are assumed to have a permanent **home connection** that never
changes. Hosts also have a permanent home address that can be used to determine their home locations. The routing goal in systems with mobile hosts is to make it possible to send packets to mobile hosts using their home addresses and have the packets efficiently reach them wherever they may be.



A WAN to which LANs, MANs, and wireless cells are attached.

In the model of Fig. 5-18, the world is divided up (geographically) into small units called areas, where an area is typically a LAN or wireless cell. Each area has one or more **foreign agents,** which are processes that keep track of all mobile hosts visiting the area. In addition, each area has a home agent, which keeps track of hosts whose home is in the area, but who are èiiif5 visiting another area. When a new host enters an area, either by connecting to it (e.g., plugging into the LAN) or just wandering into the cell, his computer must register itself with the foreign agent there. The registration procedure typically works like this:

1. Periodically, each foreign agent broadcasts a packet announcing its existence and address. A newly-arrived mobile host may wait for one of these messages, but if none arrives quickly enough, the mobile host can broadcast a packet saying: Are there any foreign agents around?

2. The mobile host registers with the foreign agent, giving its home address, current data link layer address, and some security information.

3. The foreign agent contacts the mobile host's home agent and says:

One of your hosts is over here. The message from the foreign agent to the home agent contains the foreign agent's network address. It also includes the security information

4. The home agent examines the security information, which contains a timestamp, to prove that it was generated within the past few seconds. If it is happy, it tells the foreign agent to proceed.

5. When the foreign agent gets the acknowledgement from the home agent, it makes an entry in its tables and informs the mobile host that it is flow registered.

When a host leaves an area that, too, should be announced to allow debut many users abruptly turn off their computers when done. When a packet is sent to a mobile host, it is routed to the host's home LAN as illustrated in step 1 of Fig 5-19. Here the sender, in the northwest city of Seattle, wants to send a packet to a host normally across the United States in New York. Packets sent to the mobile host on its home LAN in New York are intercepted by the home agent there. The home agent then looks up the mobile host's new (temporary) location and finds the address of the foreign agent handling the mobile host, in Los Angeles. The home agent then does two things. First, it encapsulates the packet the payload field of an outer packet and sends the latter to the foreign agent (step 2 Fig. 5-19). This mechanism is called tunneling. After getting the encapsulated packet, the foreign agent removes the original packet from the payload field and sends it to the mobile host as a data link frame. Second, the home agent tells the sender to henceforth send packets to the mobile host by encapsulating them in the payload of packets explicitly addressed to the foreign agent instead of just sending them to the mobile host's home address (step 3). Subsequent packets can now be routed directly to the host via the foreign agent (step 4), bypassing the home location entirely.

The various schemes that have been proposed differ in several ways. First' there is the issue of how much of this protocol is carried out by the routers an how much by the hosts, and in the latter
case, by which layer in the hosts. Second, in a few schemes, routers along the way record mapped addresses can intercept and redirect traffic even before it gets to the home location. in some schemes each visitor is given a unique temporary address; in others, temporary address refers to an agent that handles traffic for all visitors.

Fourth. the schemes differ in how they actually manage to arrange for packets are addressed to one destination to be delivered to a different one. One choice is changing the destination address and just retransmitting the modified packet. Alternatively, the whole packet, home address and all, can be encapsulated inside the payload of another packet sent to the temporary address. Finally,
the schemes differ in their security aspects.

2. CONGESTION CONTROL ALOGORITHM

Congestion:

- When too many packets are present in (a part of)the subnet, performance degrades. This situation is called congestion
- At very high traffic, performance collapses completely and almost no packets are delivered.

3.1 principal of congestion control

- Slow processor can also cause congestion
- The point to point traffic between the sender and receiver that a fast sender cannot continuously transmit data faster then the receiver is able to absorb it
- Solution into in to 2 groups
  1. Open loop
  2. Close loop
     1. Open loop
        - Open loop attempt to slove the problem by good design
        - Deciding when to accept file
        - Deciding when to discard packets

2. Close loop
   - Which contain 3 parts
     1. Monitor the system to detect when and where congestion occur
     2. Pass this information to places where action can be taken
     3. Adjust system operation to correct the problem

Feed back

- Transfer the information about the congestion
- Announcing the problem
- Extra packets increase load
- Take appropriate action to reduce the congestion
- Time must be adjusted carefully

3.2 Congestion prevention policies

- These systems are designed to minimize congestion in the **find** place
- They try to achieve their goal by using appropriate policies at various levels

Transport layer

Policies

- Retransmission policy
- Out of order catching poly
- Acknowledgement policy
- Flow control policy
- Time out determination

Network layer

- Packet discard policy
- Routing algorithm
- Packet life time management
- 1 Packet service policy

Data link layer

- Retransmission policy
- Out of order catching policy

- Acknowledgement catching policy
- Flow control catching policy

3.3 congestion control in virtual – circuit subnets

Two routers are congested. A wants to setup a connection to B. this connection would passs through one congestion routers to avoid this congestion

1. omitting the congested router and all of their lines
2. ----(dashed)→ shows a possible route for the virtual effective circuit that avoids the congested router

3.4 control in datagram subnets

Each newly arriving packets is checked to see if its output line is in warning if ti some action is taken.

1. **The warning bit**

- When the packet arrived at its destination the transport entity copied the bit and next acknowledgement send back to the source A
- Source then cut back on traffic
- The source monitored the fraction of acknowledgement
- The bit adjust the transmission rate according
- As long as the warning bits continued to flow in, the source continued to decrease it transmission rate
- Traffic increased when the router was in trouble

2.Chock block packets:

- It uses a round about means to tell the source to slow down
- In this approach the router sends a chock packet. Back to the packets the original packet is tagged
- So that it will not generate any more chock packet it is required to reduced the traffic

3.Hop-by- hop choke packets

Sending chock packets to the source hosts does not work well because the reaction is so slow

3.5 load shedding

When router are being inundated (busy) by packets that they cannot they just throw them away

Low priority packet cheaper to send them the high-priority once
Alternatively the sender allowed sending high- priority packets under conditions of high load. Load increased they would be discarded.

TRANSPROT LAYER:

4. **<u>ELEMENTS OF TRANSPORT PROTOCOLS</u>:**

The transport service is implemented by a **transport protocol used** between the two transport entities. In some ways, transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control. In Fig. 6-7, at the data link layer, two routes *communicate* directly via a physical channel, whereas at the transport layers this physical channel is replaced by the entire subnet.

In the data link layer, it is not necessary *for a* router to specify which router it wants to talk to—each outgoing line uniquely specifies a particular router. In the transport layer, explicit addressing of destinations is required. The process of establishing a connection over the wire of Fig. 6-7(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). In the transport layer, initial connection establishment is more complicated. Difference between the data link layer and the transport layer is the potential existence of storage capacity in the subnet. A final difference between the data link and transport layers is one of amount rather than of kind. Buffering and flow control are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer.

4.1 Addressing:

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these end points are called **ports.** In ATM networks, they are called **AAL-SAPs.** We will use the generic term **TSAP, (Transport Service Access Point).** The analogous end points in the network layer (i.e. network layer addresses) are then called **NSAPs.** IP address are examples of NSAPs.

Figure 6-8 illustrates the relationship between the NSAP, TSAP and transport connection. Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host. The purpose of having TSAP is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport end points that share that NSAP.

A possible scenario for a transport connection is as follows.

1. A time of day server process on host 2 attaches itself to TSAP *1522* to wait for an incoming call. 2. An application process on host 1 wants to find out the time-of-day, so it issues a **CONNECT** request specifying TSAP 1208 as the source and TSAP 1522 as the destination. This result in a transport connection being established between the application process on host 1 and server 1 on host 2.

3. The application process then sends over a request for the time.

4. The time server process responds with the current time.

5. The transport connection is then released.

There are other servers on host 2 that are attached to other TSAPS and waiting for incoming connections that arrive over the same NSAP. Stable TSAP addresses work for a small number of key services that never change (e.g. the Web server). If there are potentially many server processes, most of which are rarely used, it is wasteful to have each of them active and listening to a stable TSAP address all day long. In short, a better scheme is needed. One such scheme is shown in Fig. 6-9. It is known as the **initial connection protocol**. Instead of every conceivable server listening at a well-known TSAP, each machine that wishes to offer services to remote users has a special **process server** that acts as proxy for less heavily used servers. It listens to a set of ports at the same time,

waiting for a connection request. Potential users of a service begin by doing a CONNECT request, specifying the TSAP address of the service they want. If no server is waiting for them, they get a connection to the process server, as shown in Fig. 6-9(a).

After it gets the incoming request, the process server spawns the requested server. allowing it to inherit the existing connection with the user. The new server then does the requested work, while the process server goes back to listening for new requests as shown in (Refer class notes for
fig.). In this model, here exists a special process called a **name server** or sometimes a **directory server**. To find the TSAP address corresponding to a given service name, such as "time of day," a user sets up a connection to the name server (which listens to a well-known TSAP). The user then sends a message specifying the service name, and the name server sends back the TSAP address. Then the user releases the connection with the name server and establishes a new one with, the desired service. In this model, when a new service is created, it must register itself with the name server, giving both its service name (typically, an ASCII string) and its TSAP. The name server records this information in its internal database.

4.2 <u>Connection establishment:</u>

It would seem sufficient for one transport entity to just send a CONNECTION REQUEST TPDU to the destination and wait for a CONNECTION ACCEPTED reply. Imagine a subnet that is so congested that acknowledgements hardly ever get back n time and each packet times out and is retransmitted two or three times. Suppose that the subnet uses datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the subnet and take a long time to arrive, that is, they are stored in the subnet and pop out much Later.

The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person, and then releases the connection. Unfortunately, each packet in the scenario is duplicated and stored in the subnet. After the connection has been released, all the packets pop Out of the subnet and arrive at the destination in order, asking the bank to establish a new connection, transfer money (again), and release the connection. The bank has no way of telling that these are duplicates. The problem of delayed duplicates, can be handled with special emphasis on algorithms for establishing connections in a reliable way, so that nightmares like the one above cannot happen. One way is to use throwaway transport addresses. In this approach, each time a transport address is needed, a new one is generated.

When a connection is released, the address is discarded and never used again. This strategy makes the process server model of Fig. 6-9 impossible. Another possibility is to give each connection a connection identifier (i.e., a sequence number incremented for each connection established) chosen by the initiating party and put in each TPDU, including the one requesting the connection. After each connection is released, each transport entity could update a table listing Obsolete connections as (peer transport entity, connection identifier) pairs.

Whenever a Connection request comes in, it could be checked against the table, to see if t belonged to a previously-released connection. Unfortunately this scheme has a basic flaw: it requires each transport entity to maintain a certain amount of history information indefinitely. If a machine Crashes and loses its memory, it will no longer know which connection identifiers have already been used. Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted subnet design.

2. Putting a hop counter in each packet.

3. Time stamping each packet.

The first method includes any method that prevents packets from looping, combined with some way of bounding congestion delay over the (now known) longest possible path. The second method consists of having the hop count initia1ized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.

The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time.

This latter method requires the router clocks to be synchronized. Linear relation between time and initial sequence numbers is shown in Fig. 6-10. Once both transport entities have agreed on the initial sequence number, any sliding window protocol can be used for data flow control. In reality, the initial sequence number curve (shown by the heavy line) is not linear, but a staircase, since the clock advances in discrete steps. A problem occurs when a host crashes. When it comes up again, its transport entity does not know where it was in the sequence space. One solution is to require transport entities to be idle for *T*

sec after a recovery to let all old TPDUs die off. To avoid requiring T sec of dead time after a crash, it is necessary to introduce a new restriction on the use of sequence numbers. Let *T*, the maximum packet lifetime, be 60 sec and let the clock tick once per second. As shown by the heavy line in Fig. 6-10(a), the initial sequence number for a connection opened at time x will be x. Imagine that at t = 30 sec, an ordinary data TPDU being sent on (a previously opened) Connection *5* is given sequence number 80.

Call this TPDU X. Immediately after sending TPDU X, the host crashes and then quickly restarts. At t = 60, it begins reopening connections 0 through 4. At t = 70, it reopens connection *5*, initial sequence number 70 as required. Within the next 15 sec it sends data TPDUs 70 through 80. Thus, at t = *85* a new TPDU with sequence number 80 and connection *5* has been injected into the subnet. Unfortunately, TPDU X still exists. If it should arrive at the receiver before the new TPDU 80, TPDU X will be accepted and the correct TPDU 80 will be rejected as a duplicate. To prevent such problems, we must prevent sequence numbers from being used (i.e., assigned to new TPDUs) for a time *T* before their potential use as initial sequence numbers. The illegal combinations of time and sequence number are shown as the **forbidden region** in Fig. 6-10(a). Before sending any TPDU on any are connection, the transport entity must read the clock and check to see that it is not in the forbidden region.

The protocol can get itself into trouble in two distinct ways. If a host send much data too fast on a newly-opened connection, the actual sequence number versus time curve may rise more steeply. This means that the maximum data rate on any connection is one TPDU per clock tick. It also means that the transport entity must wait until clock ticks before opening a new connection after a crash restart, lest the same number be used twice. From Fig. 6-10(b), we see that at any data rate less than the clock rate; the curve of actual sequence numbers used versus time will eventually run into the forbidden region from the left.

The greater the slope of the actual sequence number curve, the longer this event will be delayed. As we stated above, just before sending every TPDU, the transport entity must check to see if it is about to enter the forbidden region, and if so, either delay the TPDU for *T* sec or resynchronize the sequence numbers.

The clock-based method solves the delayed duplicate problem for data TPDUs, but for this method to be useful, a connection must first be established. Since control TPDUs may also be delayed, there is a potential problem in getting both sides to agree- on the initial sequence number. To solve this problem, Tomlinson *(1975)* introduced the **three-way hand-shake**. This establishment protocol does not require both sides to begin sending with the same sequence number, so it can be used with synchronization methods other than the global clock method.

The normal setup procedure when host 1 initiates is shown in Fig.. 6-11(a). Host 1 chooses a sequence number, x, and sends a CONNECTION REQUEST TPDU containing it to host 2. Host 2 replies with an ACK TPDU acknowledging x and announcing its own initial sequence number, y. Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data TPDU that it sends. The three-way handshake works in the presence of delayed duplicate control TPDUs. In Fig. 6-11(b), the first TPDU is a delayed duplicate CONNECTION REQUEST from an old connection. This TPDU arrives at host 2 without host l's knowledge. Host 2 reacts to this TPDU by sending host 1 an ACK TPDU, in effect asking for verification that host 1 was indeed trying to set a new connection When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage. The Worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet. This case is shown in Fig. 6-11(c). As in the previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it. At this point it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing that no TPDUs containing sequence number y or acknowledgements to y are still in existence. When the second delayed TPDU arrives at host 2, the fact that z been acknowledged rather than y tells host 2 that this, too, is an old duplicate.

4.3 Connection release:

Releasing a connection is easier than establishing one. There are two styles of terminating a connection: asymmetric release and symmetric release. Asymmetric release is the way the telephone system works: when one parry hangs up the connection is broken. Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately. Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. 6-12. After the connection is established, host 1 sends a TPDU that arrives properly at host 2. Then host 1 sends another TPDU. Unfortunately, host 2 issues a DISCONNECT before the second TPDU arrives. The result is that the connection is released and data are lost. A more sophisticated release protocol is needed to avoid data loss. One way is to use symmetric release, in which each direction is release in-pendently of the other one. Here, a host can continue to receive data even after it

has sent a DISCONNECT TPDU. Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. Unfortunately this protocol does not always work. There is a famous problem that illustrates this issue. It is called the t**wo-army problem.** we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) TPDU to initiate the connection release. When it arrives, the recipient sends back a DR TPDU, too, and starts a timer, just in case its DR is lost. When this DR arrives, the original sender sends back an ACK TPDU and releases the connection. Finally, when the ACK TPDU arrives, the receiver also releases the connection. Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow. This action is different from a transport user issuing a DISCONNECT primitive.

If the final ACK TPDU is lost, the situation is save by the timer. When the timer expires, the connection is released anyway. Now consider the case of the second DR being lost. The user initiating the disconnection will not receive the expected response, will time out, and will start all over again. we see how this works, assuming that the second time no TPDUs are lost and all TPDUs are delivered correctly and on time. Our last scenario, Fig. 6-14(d), is the same as Fig. 6-14(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost TPDU's. After *N* retries, the sender just gives up and releases the connection. Meanwhile, the receiver times out and also exits. While this protocol usually suffices, in theory it can fail if the initial DR and *N* retransmissions are all lost. The sender will give up and release the connection, while the other side knows nothing at all about the attempts to disconnect and is still fully active. This situation results in a half-open connection. We could have avoided this problem by not allowing the sender to give up after *N* retries but forcing it to go on forever until it gets a response. However, if the other side is allowed to time out, then the sender will indeed go on forever, because no response will ever be forthcoming. If we do not allow the receiving Side to time out, then the protocol hangs in Fig. 6-14(d). One way to kill off half-open connections is to have a rule saying that if no TPDU's have arrived for a certain number of seconds, the connection is then automatically disconnected. The other side will detect the lack of activity and also disconnect. Of course, if this rule is introduced, it is necessary for each transport entity to have a timer that is stopped and then restarted whenever a TPDU is sent. If this timer expires, a dummy and transmitted, just to keep the other side from disconnecting. On the other hand, if the automatic disconnect rule is used and too many dummy TPDU5 in a row are lost on an otherwise idle connection, first one side, then the other side will automatically disconnect.

<u>4.4 Flow control and buffering:</u>

One of the key issues has come up before: flow control. In some ways the flow control problem in the transport layer is the same as in the data link layer, but in other ways it is different. The basic similarity is that in both layers a sliding window or other scheme is needed on each connection to keep a fast transmitter from overrunning a slow receiver. In the data link layer, the sending side must buffer outgoing frames because they might have to be retransmitted. If the subnet provides datagram service, the sending transport entity must also buffer. If the receiver knows that the sender buffers all TPDUs until they are acknowledged, the receiver may or may not dedicate specific buffers to specific connections. The receiver may, for example, maintain a single buffer pool shared by all connections.

When a TPDU comes in, an attempt is made to dynamically acquire a new buffer. If one is available, the TPDU is accepted; otherwise, it is discarded. Since the sender is prepared to retransmit TPDUs lost by the subnet, no harm is done by having the receiver drop TPDUs, although some resources are wasted. The sender just keeps trying until it gets an acknowledgement. If the network service is unreliable, the sender must buffer all TPDUs sent, just as in the data link layer. If the sender knows that the receiver always has buffer space, it need not retain copies of the TPDUs it sends. However, if the receiver cannot guarantee that every incoming TPDU will be accepted the sender will have to buffer anyway. In the latter case, the sender the network layer's acknowledgement, because the acknowledgement means only that the TPDU arrived, not that it was accepted. If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identically-sized buffers, with one TPDU per buffer.However, if there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, a pool of fixed-sized buffers presents problems. If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives. If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs, with the attendant complexity. Another approach to the buffer size problem is to use variable-sized buffers, as

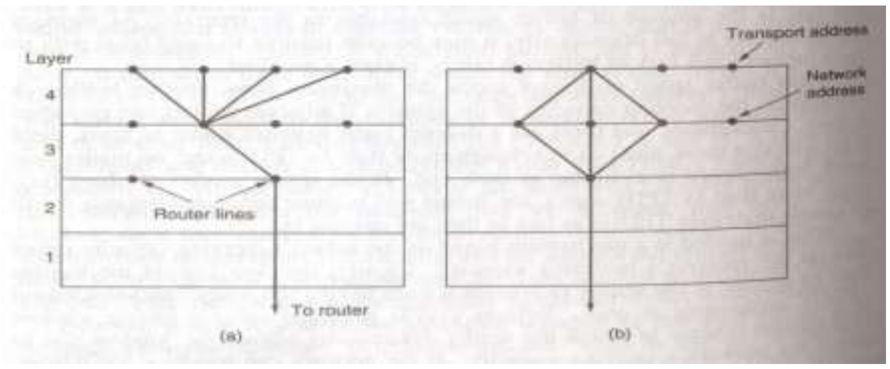
in (refer class notes for Fig.). The advantage here is better memory utilization, at the price of more complicated buffer management. A third possibility is to dedicate a single large circular buffer per connection, as in Fig. 6-15(c). This system also makes 0od use of memory; provided that all connections are heavily loaded, but are poor if some connections are lightly loaded. The optimum trade-off between source buffering and destination buffering depends on the type of traffic carried by the connection. For low-bandwidth bursty traffic, such as that produced by an interactive terminal, it is better not to dedicate any buffers, but rather to acquire them dynamically at both ends the sender cannot be sure the receiver will be able to acquire a buffer, the Since must retain a copy of the TPDU until it is acknowledged. On the other h sender and, for file transfer and other high-bandwidth traffic, it is better if the receiver does dedicate a full window of buffers, to allow the data to flow at maximum speed. Thus, for low-bandwidth bursty traffic, it is better to buffer at the sender, and for high bandwidth smooth traffic, it is better to buffer at the receiver.

To manage dynamic buffer allocation is to decouple the buffering from the acknowledgements. Dynamic buffer management means, in effect, a variable-sized window, Figure 6-16 shows an example of how dynamic window management might work in a datagram subnet with 4-bit sequence numbers. Assume that buffer allocation information travels in separate TPDUs, as shown, and is not piggybacked onto reverse traffic. Initially, *A* wants eight buffers, but is granted only four of these. It then sends three TPDUs, of which the third is lost. TPDU 6 acknowl edges receipt of all TPDUs up to and including sequence number 1, thus allowing *A* to release those buffers, and furthermore informs *A* that it has permission to send three more TPDUs starting beyond 1 (i.e., TPDUs 2, 3, and 4). *A* knows that it has already sent number 2, so it thinks that it may send TPDUs 3 and 4, which it proceeds to do. At this point it is blocked and must wait for more buffer allocation. Timeout-induced retransmissions (line 9), however, may occur while blocked, since they use buffers that have already been allocated. In line 10, *B* acknowledges receipt of all TPDUs up to and including 4 but refuses to let *A* continue. Such a situation is impossible with the fixed window protocols. The next TPDU from *B* to *A* allocates another buffer and allows *A* to continue.

Potential problems with buffer allocation schemes of this kind can arise in datagram networks if control TPDUs can get lost. Look at line 16. *B has now* allocated more buffers to *A*, but the allocation TPDU was lost. Since TPDUs are not sequenced or timed out, *A* is now deadlocked. To prevent this situation, each host should periodically send control TPDUs giving the acknowledgement and buffer status on each connection. That way, the deadlock will be broken, sooner or later.

4.5 Multiplexing:

Multiplexing several conversations onto connections, virtual circuits, and



physical links plays a role in several layers of the network architecture. In the transport layer the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it. When a TPDU comes in, some way is needed to tell which process to give it to. When a TPDU comes in, some way is needed to tell which machine has to give it to. This situation, called **upward multiplexing,** is shown in Fig. 6.l7(a). in this figure, four distinct transport connection all use the same network connection (e.g., IP address) to the remote host. z**(a) Upward multiplexing. (b) Downward multiplexing.**

Multiplexing can also be useful in the transport layer. If a user needs more bandwidth than one virtual circuit can provide, a way out to open multiple network connections and distribute the traffic among them a round-robin basis.This modus operandi is called **downward multiplexing.**

4.6 Crash recovery:

If hosts and routers are subject to crashes, recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. The latter ones can be retransmitted. A more troublesome problem is how to recover from host crashes. In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot. Let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol. The transport layer on the server simply passes the incoming TPDUs to the transport user, one by one. Partway through the transmission, the server crashes. When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was. In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections. Each client can be in one of two states: one TPDU outstanding, S1, or no TPDUs outstanding, *S0.* Based on Only this state information, the client must decide whether to retransmit the most recent TPDU.

The client should retransmit only if and only if it has an unacknowledged TPDU outstanding (i.e., is in state *S1)* when it learns of the crash. For example, the situation in which the server's transport entity first sends an acknowledgement, and then, when the acknowledgement has been, writes to the application process. Writing a TPDU onto the output stream and sending an acknowledgement are two distinct events that cannot be done simultaneously. If a crash occurs after the acknowledgement has been sent but before the write has been done, the client will receive the acknowledgment and thus be in state *S0* when the crash recovery announcement arrives. The client will therefore not retransmit, (incorrectly) thinking that the TPDU has arrived.

This decision by the client leads to a missing TPDU. Three events are possible at the server: sending an acknowledgement *(A)*, writing to the output process *(W)*, and crashing *(C)*. The three events can occur in six different orderings: *AC(W), AWC, C(AW), C(WA), WAC,* and *WC(A),* where the parentheses are used to indicate that neither *A* nor *W* can follow *C* (i.e., once it has crashed, it has crashed). Figure 6-18 shows all eight combinations of client and server strategy and the valid event sequences for each one. For each strategy there is some sequence of events that causes the protocol to fail. For example, if the client always retransmits, the *AWC* event will generate an undetected duplicate, even though the other two events work properly.

5. <u>THE INTERNET TRANSPORT PROTOCOLS: TCP</u>

Introduction of the TCP

TCP (Transmission control protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internet work .An internet work differs from a single network because different parts may have widely different topologies, bandwidths, delays, packet sizes, and other parameters.

The TCP service model

TCP service is obtained by both the sender and the receiver creating end points, called sockets .Each socket has a socket number consisting of the IP address of the IP address of the host and 16-bit number local to that host, called a port.

Port numbers below 1024 are called well known ports and are reserved for standard services.For example,any process wishing to establish a connection to a host to transfer a file using FTP can connect to establish a connection to a host transfer a file using FTP can connect to the destination host’s port 21 to contact its FTP daemon.

The TCP protocol

A key feature of TCP ,and one which dominates the protocol design, is that every byte on the TCP connection has its own 32-bit sequence number.
The sending and receiving TCP entities exchange data in the form of segments. A TCP segments consists of a fixed 20 byte header followed by zero or more data bytes.

The TCP Segment header.

The sequence number and Acknowledgement number fields perform their usual functions. Note that the latter specifies the next byte accepted, not the last byte correctly received.

TCP Connection Establishment.

Same like as connection establishment in the transport protocol.

TCP connection Management Modeling.

Each connection starts in the CLOSED state. It leaves that state when it does either a passive open(LISTEN),or an active open(CONNECT).If the other side does the opposite one, a connection is established and the state becomes ESTABLISHED. Connection release can be initiated by either side. When it is complete, the state returns to CLOSED.

STATE

1.CLOSED

2.LISTEN

3.SYN RCVD

4.SYN SENT

5.ESTABLISHED

6.FIN WAIT 1

7.FIN WAIT 2

8.TIMED WAIT

9.CLOSING

10.CLOSE WAIT

11.LAST ACK

TCP transmission policy

Suppose the receiver has a 4096-byte buffer. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment.

TCP Congestion Control

The internet solution is to realize that two potential problem exists. Network capacity and receiver capacity and to deal with each of them separately. To do so, each sender maintains two windows: the window the receiver has granted and a second window, the congestion window.

Let us look at the internet congestion control algorithm. It uses a third parameter, the threshold, initially 64KB, in addition to the receiver and congestion windows.

TCP Timer Management

TCP uses multiple timers to do its work.The most important of these is the retransmission timer.When a segment is sent,a retransmission timer is started.If the segment is acknowledgement before the timer expires,the timer is stopped.

Keep alive timer

When a connection has been idle for a long time,the keep alive timer may go off to cause one side to check whether the other side is still there.If it fails to respond,the connection is terminated.This features is controversial it adds overhead and may terminate an otherwise healthy connection due to a transient network partition.

POINTS TO REMEMBER

- The Transport layer Job is routing packets from the source to the destination.
- Types of routings-hierarchical routing, routing for mobile hosts, broadcast routing, multicast routing and routing in peer-to-peer networks.
- The new version of IP is IPv6.
- The internet has to protocols two main protocols: UDP and TCP.
- TCP –It provides reliable bidirectional byte stream.
- Multiplexing and demultiplexing multiple processes using a single IP address.
- UDP can be used for client-server interaction
- TCP can be used for client-server interaction with a reduced number of packets.

Expected questions

Section –A

1.The _________ layer is the key to understand layered protocols.

2. what are two main protocols in Internet ?

3. Define threshold.

4. What is meant by slow start?

5. Expand RTCP

Section-B

1. Explain briefly about store-and-forward techniques?
2. Explain briefly about the optimality principle?
3. Explain flooding?
4. Explain about routing mobile hosts?
5. Explain about Berkeley sockets?

Section -c

1. Comparison of virtual-circuit and datagram subnets?
2. Explain about shortest path routing with algorithm?
3. Explain about distance vector routing?
4. Explain transport service primitives?
5. Explain about flow control and buffering?
6. Write notes on elements of transport layer?

UNIT V: **APPLICATION LAYER:** DNS – E-mail. NETWORK SECURITY: Cryptography – Symmetric Key Algorithms – Public Key Algorithms – Digital Signatures.

OBJECTIVES:

To learn about

- Role of DNS over Internet.
- Functioning of Email.
- Types of algorithms used for Cryptography.
- Cryptography Roles in Network security.
- Accessing of Digital signature over network.

THE APPLICATION LAYER:

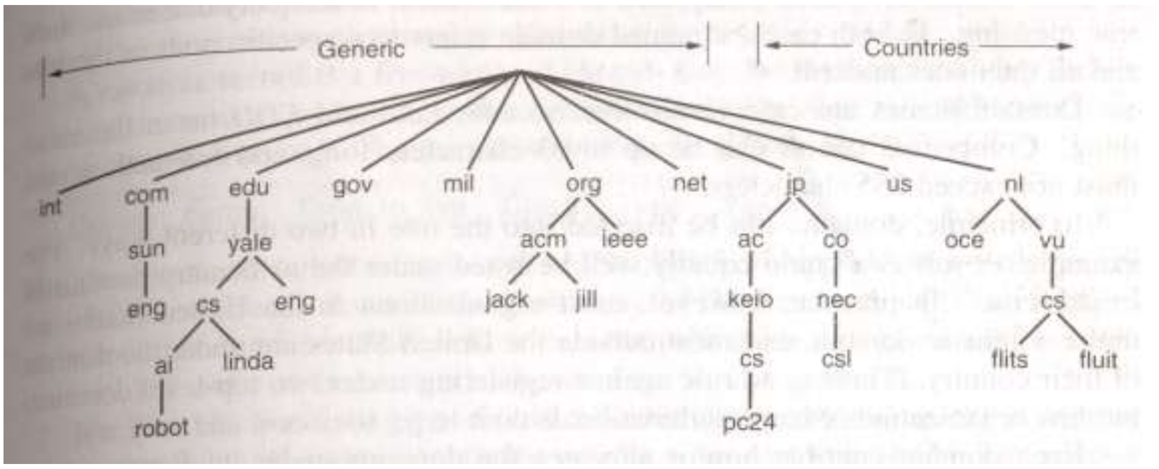
1. <u>THE DOMAIN NAME SYSTEM (DNS):</u>

The programs theoretically could refer to hosts, mailboxes, and other resources by their network (e.g., IP) addresses, these addresses are hard for people remember. Also, sending e-mail to *tana@128.I1I.24.41* means that if Tana's ISP or organization moves the mail server to a different machine with a different address, her e-mail address has to change. Consequently, ASCII names were introduced to decouple machine names from machine addresses. In this way, Tana's address be like The network itself understands only numerical addresses, so some mechanism is required the ASCII strings to network addresses. Way back in the ARPANET, there was simply a file, *hosts.txt* that listed all the hosts and their IP addresses. Every night, all the hosts would fetch it from the site at which it was maintained.

The essence of DNS is the invention of a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme It is primarily used for mapping host names and e-mail destinations to IP addresses but can also be used for other purposes. DNS is defined in RFC8 1034 and *1035.* To map a name onto an IP address, an application program calls a library procedure called the resolver, passing it the name as a parameter. The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to the resolver, which then returns it to the caller. Armed with the IP address, the program can then establish a TCP connection with the destination or send it UDP packets.

1.1 The DNS Name Space:

Managing a large and constantly changing set of names is a nontrivial problem. In the postal system, name management is done by requiring letters to specify (implicitly or explicitly) the country, state or province, city, and street address of the addressee, company and contain thousands of hosts. The Internet is divided into over 200 top-level domains, where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned, and so on. All these domains can be represented by a tree, as shown in Fig. The leaves of the tree represent domains that have no subdomains (but do contain machines, of course). A leaf domain may contain a single host, or it may represent a company and contain thousands of hosts. The top-level domains come in two flavors: generic and countries. The original generic domains were *corn (commercial), edu* (educational institutions), gov (the U.S. Federal Government), in*t* (certain international organizations), mil (the U.S armed forces), *net* (network providers), and *org* (nonprofit organization). The country domains include one entry for every country, as defined in ISO 3166.



A portion of the Internet domain name space.

In November 2000, ICANN approved four new, general-purpose, top-level domain, namely, *biz* (businesses), *info* (information), *name* (people's names), and *pro* (professions, such as doctors and lawyers). In addition, three more specialized top-level domains were introduced at the request of certain industries. These are *aero* (aerospace industry), *coop* (co-operatives), and *museum* (museums). Other top-level domains will be added in the future. In general, getting a second-level domain, such as *narne-of-company.com,* is easy. It merely requires going to a registrar for the corresponding top-level domain *(corn* in this case) to check if the desired name is available and not somebody else's trademark. If there are no problems, the requester pays a small annual fee and gets the name. By now, virtually every common (English) word has been taken in the *corn* domain. Try household articles, animals, plants, body parts, etc. Nearly all are taken. Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced "dot"). Thus, the engineering department at Sun Microsystems might be *eng.sun.corn.,* rather than a UNIX-style name such as */com/sun/eng* Notice that this hierarchical naming means that *eng.s* does not conflict with a potential use of *eng* in *eng.yale.edu.,* which mig be used by the Yale English department. Domain names can be either absolute or relative. An absolute domain name always ends with a period (e.g.,*eng.sun.com.),* whereas a relative one does not.Relative names have to be interpreted in some context to uniquely determine their true meaning. In both cases, a named domain refers to a specific node in the tree and all the nodes under it. Domain names are case insensitive, so *edu, Edu,* and *EDU* mean the same thing. Component names can be up to 63 characters long, and full path names must not exceed *255* characters. In principle, domains can be inserted into the tree in two different ways For example, *cs.vale.edu* could equally well be listed under the *us* Country domain as *cs.yale.ct.us.* In practice, however, most organizations in the United States are under a generic domain, and most outside the United States are under the domain of their country. There is no rule against registering under two top-level domains, but few organizations except multinationals do it (e.g., *son y.corn* and *son v.nl).* Each domain controls how it allocates the domains under it. For example, Japan has domains *ac.jp* and *co.jp* that mirror *edu* and *corn.* The Netherlands does not make this distinction and puts all organizations directly under nl. Thus, all three of the following are university computer science departments:

1. *cs.yale.edu* (Yale University, in the United States)
2. *cx. vu.nl* (Vrije Universiteit, in The Netherlands)
3. *cs.keio.ac.jp* (Keio University, in Japan)

To create a new domain, permission is required of the domain in which it will be included. For example, if a VLSI group is started at Yale and wants to be known as *vlsi.cs.vale.edu,* it has to get permission from whoever manages *cs.yale.edu.* Similarly, if a new university is chartered, say, the University of Northern South Dakota, it must ask the manager of the *edu* domain to assign **it** *unsd.edu.* In this way, name conflicts are avoided and each domain can keep track of all its subdomains. Once a new domain has been created and registered, it can create subdomains, such as *cs.unsd.edu,* without getting permission from anybody higher up the tree. Naming follows organizational boundaries, not physical networks. For example, if the computer science and electrical engineering departments are located in the same building and share the same LAN, they can nevertheless have distinct domains. Similarly, even if computer science is split over Babbage Hall and Turing Hall, the hosts in both buildings will normally belong to the same domain.

<u>1.2 Resource records:</u>

Every domain, whether it is a single host or a top-level domain, can have a set of resource records associated with it. For a single host, the most common resource record is just its IP address, but many other kinds of resource records exist. When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus, the primary function of DNS is re domain names onto resource records.

A resource record is a five-tuple. Although they are encoded in binary for efficiency, in most expositions resource records are presented as ASCII text, one line per resource record. The format we will use is as follows:

Domain_name Time_to_live Class Type Value

The Domain name tells the domain to which this record applies. Many records exist for each domain and each copy of the database holds information about multiple domains. This field is thus the primary search key used to satisfy queries The order of the records in the database is not significant.

The ***Time_to_live*** field gives an indication of how stable the record is. Information that is highly stable is assigned a large value, such as 86400 (the number of seconds in I day). Information that is highly volatile is assigned a small value, such as 60 (1 minute).

The third field of every resource record is the ***Class***. For Internet information, it is always *IN.* For non-Internet information, other codes can be used, but in practice, these are rarely seen.

The *Type* field tells what kind of record this is.

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone.
A	IP address of a host	32-bit integer

MX	Mail exchange	Priority, domain willing to accept e-mail.
NS	Name Server	Name of a server for this domain.
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host Description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text.

The principal DNS resource record types for IPv4.

An *SOA* record provides the name of the primary source of information about name server's zone the e-mail address of its administrator, a unique serial number, and various flags and timeouts. The most important record type is the *A* (Address) record. It holds a 32-bit IP address for some host. Every Internet host must have at least one IP address so at other machines can communicate with it. Some hosts have two or more network connections in which case they will have one type *A* resource record per network connection. DNS can be configured to cycle through these, returning the first record on the first request, the second record on the second request, and so on. The next most important record type is the *MX* record. It specifies the name of the host prepared to accept e-mail for the specified domain. It is used because not every machine is prepared to accept e-mail. If someone wants to send e-rnail to, for example, *bill@microsoft.com,* the sending host needs to find a mail at, *microsoft.coin* that is willing to accept e-mail. The *MX* record can provide this information.

The *NS* records specify name servers. For example, every DNS database normally has an *NS* record for each of the top-level domains, so, for example, e-mail can be sent to distant parts of the naming tree. *CNAME* records allow aliases to be created. For example, a person familiar with Internet naming in general and wanting to send a message to Someone whose login name is *paul* in the computer science department at M.I.T. might guess that *paul@cs.rnit.edu* will work. Actually, this address will not work, because the domain for M.l.T.'s computer science department is *lcs.mit.edu. A*s a service to people who do not know this, M.I.T. could create a *CNAME* entry to point people and programs in the right direction. An entry like this one might do the job:

cs.mit.edu 86400 IN CNAME lcs.mt.edu

CNAME, PTR points to another name. However, unlike *CNAME,* which is really just a macro definition, *PTR* is a regular DNS datatype whose interpretation depends on the context in which it is found. It is nearly always used to associate a name with an IP address to allow lookups of the IP address and return the name of the corresponding machine. These are called **reverse lookups.** *HINFO* records allow people to find out what kind of machine and operating system a domain corresponds to. Finally, *TXT* records allow domains to identify themselves in arbitrary ways. Both of these record types are for user convenience. Finally, we have the *Value* field. This field can be a number, a domain name, or an ASCII string. The semantics depend on the record type. A short description of the *Value* fields for each of the principal record types.

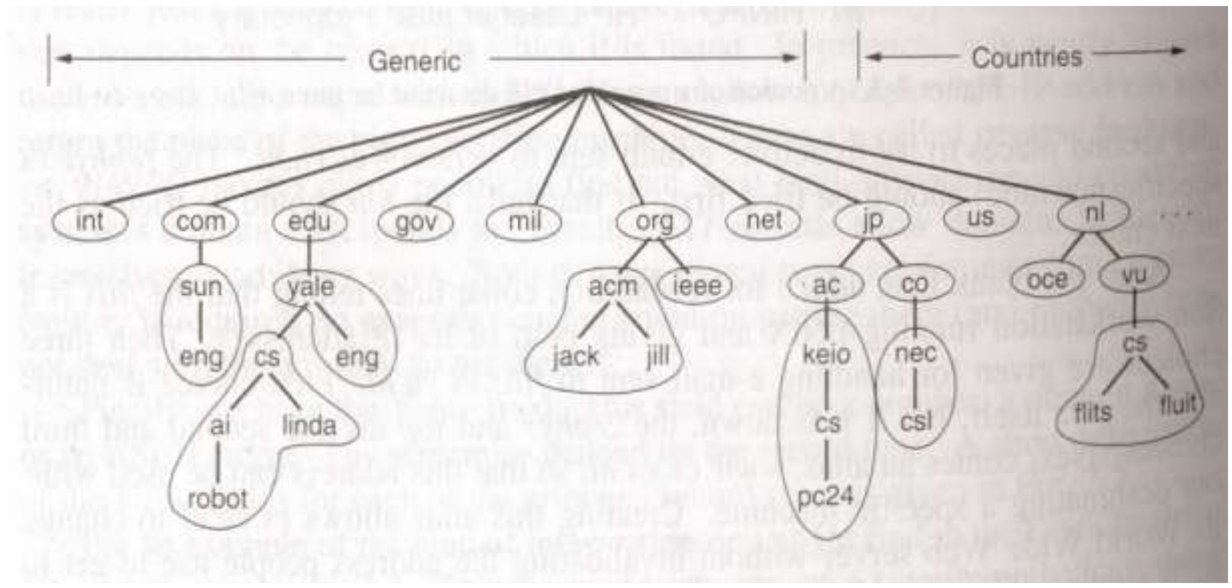
<u>1.3 Name Servers:</u>

In theory at least, a single name server could contain the entire DNS database and respond to all queries about it. In practice, this server would be so overloaded as to be useless. If it ever went down, the entire Internet would be crippled.

<u>ZONES</u>

To avoid the problems associated with having only a single source of information, the DNS name space is divided into non overlapping **zones.** One possible way to divide the name space of Fig. 7-I is shown in Fig. 7-4. Each zone contains some part of the tree and also contains name servers holding the information about that zone. Normally, a zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their

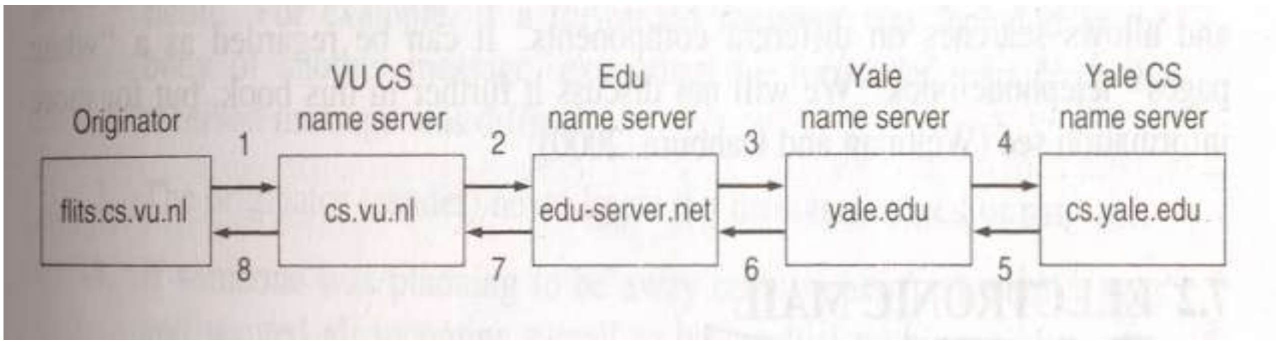
information from the primary name server. To improve reliability, some servers for a zone can be located outside the zone



Part of the DNS name space showing the division into zones.

Where the zone boundaries are placed within a zone is up to that zone's administrator. This decision is made in large part based on how many name servers are desired, and where. For example, in Fig. 7-4. Yale has a server for its that handles *eng.yale.edu* but not *cs.yale.edu,* which is a separate zone with its servers. Such a decision might be made when a department such as not wish to run its own name server, but a department such as come does. Consequently, *cs.yale.edu* is a separate zone but *eng.yale.edu* is not.

When a resolver has a query about a domain name, it passes the query to one of the local name servers. If the domain being sought falls under the jurisdiction of the name server, such as *ai.cs.yale.edu* falling under *cs.yule.edu,* it returns the authoritive resource records. An authoritative **record** is one that comes from the authority that manages the record and is thus always correct. Authoritative records are in contrast to cached records, which may be out of date. The domain is remote and no information about the requested domain is available locally, the name server sends a query message to the top-level name server for the domain requested. To make this process clearer, consider the example of Fig. 7-5. Here, a resolver on flits*.cs.vu.nI* wants to know the IP address of the host *Iindu.cs.yale.edu.* In step 1, it sends a query to the local name server, *cs.vu.nI.* This query contains the domain name sought, the type *(A)* and the class *(IN).*



How a resolver looks up a remote name in eight steps.

Let us suppose the local name server has never had a query for this domain before and knows nothing about it. It may ask a few others nearby name servers, but if none of them know, it sends a UDP packet to the server for *edu* given in its database (see Fig. 7-5), *edu-server.net.* it is unlikely that this server knows the address of *lindu.cs.yale.edu,* and probably does not know *cs.vale.edu* either, but it must know all of its own children, so it forwards the request to the name server for *yale.edu,* (step 3). In turn, this one forwards the request to *cs.yale.edu* (step 4), which must have the authoritative resource records. Since each request is from a Client to a server, the resource record requested works its way back in steps *5* through 8. Once these records get back to the *cs.vu.nI* name server, they will be entered into a cache there, in case they are needed later. However, this information is not authoritative since changes made at *cs.yale.edu* will not be propagated to all the caches in the world that may know about for this reason. Cache entries should not live too long. This is the reason that the *Tirne_ to_live* field is included in each resource record. It tells remote name servers how long to cache records. If a certain 0 machine has had the same IP address for years, it may be safe to cache that information for 1 day. For more volatile information, it might be safer to purge the records after a few seconds or a minute. It is worth mentioning that the query method described here is known **recursive query**. Since each server that does not have the requested information goes and finds it somewhere, then reports back. An alternative form is also possible. In this form, when a query cannot be satisfied locally, the query fails, but the name of the next server along the line to try is returned. Some servers do not implement recursive queries and always return the name of the next server to try. It is also worth pointing out that when a DNS client fails to get a response before its timer goes off, it normally will try another server next time. The assumption here is that the server is probably down, rather than that the request or reply got lost. While DNS is extremely important to the correct functioning of the Internet, all it really does is map symbolic names for machines onto their IP addresses, it does not help locate people, resources, services, or objects in general. For locating these things, another directory service has been defined, called LDAP (**Lightweight Directory Access Protocol).** It is a simplified version of the OSI X.500 directory service and is described in RFC 225 1. It organizes information as a tree and allows searches on different components. It can be regarded as a "white pages" telephone book.

2. ELECTRONIC MAIL:

Electronic mail, or e-mail, as it is known to its many fans, has been around for over two decades. Before 1990, it was mostly used in academia. During the 1990s, it became known to the public at large and grew exponentially to the point where the number of e-mails sent per day now is vastly more than the number of **snail_mail** (i.e., paper) letters. E-mail, like most other forms of communication, has its own conventions and styles. In particular, it is very informal and has low threshold of use. People who would never dream of calling up or even writing a letter to a very important person do not hesitate for a second to send a sloppily-written e-mail.
E-mail is full of jargon such as BTW (By The Way), ROTFL (Rolling on The Floor Laughing), and IMHO (In My Humble Opinion). Many people also use little ASCII symbols called smile or emoticons in their e-mail. The first e-mail systems simply consisted of file transfer protocols, with the convention that the first line of each message recipient's address. As time went on, the limitations of this approach became more obvious.
Some of the complaints were as follows:

1. Sending a message to a group of people was inconvenient. Managers often need this facility to send memos to all their subordinates.
2. Messages had no internal structure, making computer processing difficult. For example, if a forwarded message was included in the body of another message, extracting the forwarded part from the received message was difficult.
3. The originator (sender) never knew if a message arrived or not.
4. If someone was planning to be away on business for several weeks and wanted all incoming e-mail to be handled by his secretary, this was not easy to arrange.

5. The user interface was poorly integrated with the transmission system requiring users first to edit a file, then leave the editor and invoke the file transfer program.

6. It was not possible to create and send messages containing a mixture of text, drawings, facsimile, and voice.

As experience was gained, more elaborate e-mail Systems were proposed. In 1982, the ARPANET e-mail proposals were published as RFC 821 (transmission Protocol) and RFC 822 (message format). Minor revisions, RFC 2821 and RFC 22, have become Internet standards, but everyone still refers to Internet e-mail as RFC 822. In 1984, CCITT drafted its X.400 recommendation. After two decades of competition e-mail systems based on RFC 822 are widely used, whereas those based on X.400 have disappeared. RFC 822-based e-mail System and a supposedly truly wonderful but nonworking, X.400 e-mail system, most organizations chose the former.

2.1 Architecture and Services:

They normally consist of two subsystems: the user agent who allow people to read and send e-mail, and the message transfer agents which move the messages from the source to the destination. The user agents are local programs that provide a command-based, menu- based, or graphical method for interacting with the e-mail system. The message transfer agents are typically system daemons, that is, processes that run in the background. Their job is to move e-mail through the system. Typically, e-mail systems support five basic functions.

- **Composition** refers to the process of creating messages and answers. Although any text editor can be used for the body of the message, the system itself can provide assistance with addressing and the numerous header fields attached to each message. For example, when answering a message, the e-mail system can extract the originator's address from the incoming e-mail and automatically insert it into the proper place in the reply.
- **Transfer** refers to moving messages from the originator to the recipient. In large part, this requires establishing a connection to the destination or some intermediate machine, outputting the message, and releasing the connection. The email system should do this automatically, without bothering the user.

- **Reporting** has to do with telling the originator what happened to the message. Was it delivered? Was it rejected? Was it lost? Numerous applications exist in which confirmation of delivery is important and may even have legal significance.
- **Displaying** incoming messages is needed so people can read their e-mail Sometimes conversion is required or a special viewer must be invoked, if then message is a PostScript file or digitized voice. Simple conversion formatting are sometimes attempted as well.
- **Disposition** is the final step and concerns what the recipient does with the message after receiving it. Possibilities include throwing it away before read'° throwing it away after reading, saving it, and so on. It should also be possible to retrieve and reread saved messages, forward them, or process them other ways. In addition to these basic services, some e-mail systems, especially internal corporate ones, provide a variety of advanced features.

When people move or when they are away for some period of time, may want their e-mail forwarded, so the system should be able to do this automatically. Most systems allow users to create **mailboxes** to store incoming e-mail.

Commands are needed to create and destroy mailboxes, inspect the contents of mailboxes, insert and delete messages from mailboxes, and so on. Corporate managers often need to send a message to each of their subordinates customers, or suppliers. This gives rise to the idea of a **mailing** list, which a list of e-mail addresses. When a message is sent to the mailing list identical copies are delivered to everyone on the list.
Other advanced features are carbon copies, blind carbon copies, high-priority e-mail, secret (i.e., encrypted) e-mail, alternative recipients if the primary one is not currently available, and the ability for secretaries to read and answer their bosses' e-mail. E-mail is now widely used within industry for intra company communication. It allows far-flung employees to cooperate on complex projects, even over many time zones. By eliminating most cues associated with rank, age, and gender, email debates tend to focus on ideas, not on corporate status. With e-mail, a brilliant idea from a summer student can have more impact than a dumb one from an executive vice president.

A key idea in e-mail systems is the distinction between the envelope and its contents. The envelope encapsulates the message. It contains all the information needed for transporting the message, such as the destination address, priority, and security level, all of which are distinct from the message itself. The message transport agents use the envelope for routing, just as the post office does. The message inside the envelope consists of two parts: the header and the body. The header contains control information for the user agents. The body is entirely for the human recipient.

2.2 <u>The User Agent:</u>

E-mail systems have two basic parts: The user agents and the message transfer agents. A user agent is a program (sometimes called a mail reader) that accepts as for commands for composing, receiving, and replying to messages, as well manipulating mailboxes. Some user agents have a fancy menu- or icon driven interface that requires a mouse, whereas others expect 1-character commands from the keyboard. Functionally, these are the same. Some systems are menu or icon-driven but also have keyboard shortcuts.

<u>Sending E-Mail:</u>

To send an e-mail message, a user must provide the message, the destination address, and possibly some other parameters. The message can be produced with a free-standing text editor, a word processing program, or possibly with a specialized text editor built into the user agent. The destination address must be in a format that the user agent can deal with. Many user agents expect addresses of the form *user@dns-address.* They are composed of *attribute* = *value* pairs separated by slashes. This address specifies a country, state, locality, personal address and a common name (Ken Smith). Many other attributes are possible so you can send to someone whose exact e-mail address you do not know, provided you enough other attributes.

<u>Reading E-Mail:</u>

When a user agent is started up, it looks at the user's mailbox for incoming e-mail before displaying anything on the screen. Then it may announce the number of messages in the mailbox or display a one-line summary of each one and wait for command.

#	Flags	Bytes	Sender	Subject
1	K	1030	asw	Changes to MINIX
2	KA	6348	trudy	Not all Trudy's are nasty
3	K F	4519	Amy N.Wong	Request for information
4		1236	bal	Bioinformatics
5		104110	kaashoek	Material on peer-to-peer
6		1223	Frank	Re:Will you review a grant proposal
7		3110	guido	Our paper has been accepted
8		1204	dmr	Re:My student's visit

Each line of the display contains several fields extracted from the envelop or header of the corresponding message. In a simple e-mail system, the choice of fields displayed is build into the program. Then user can specify which fields are to be displayed by providing a user profile, a file describing the display format. In this basic example, the first field is the message number. The second field, *Flags* can contain a *K,* meaning that the message not new but was read previously and kept in the mailbox; an *A,* meaning that the message has already been answered; and/or an *F,* meaning that the message has been forwarded to someone else. Other flags are also possible. The third field tells how long the message is, and the fourth one tells who sends the message. Since this field is simply extracted from the message, this field sent contain first names, full names, initials, login names, or whatever else the sender chooses to put there. Finally, the *Subject* field gives a brief summary of what message is about. People who fail to include a *Subject* field often discover that responses to their e-mail tend not to get the highest priority. After the headers have been displayed, the user can perform any of several actions, such as displaying a message, deleting a message, and so on. The Older systems were text based and typically used one-character commands for performing these tasks, such as T (type message), A (answer message), D (delete message), and F (forward message). More recent systems use graphical interfaces. Usually, the user selects a message with the mouse and then clicks on an icon to type, answer, delete, or forward it. E-mail has come a long way from the days when it was just file transfer. User agents make managing a large volume of e-mail. For people who receive and send thousands of messages a year, such tools are invaluable.

2.3 Message Formats

RFC 822 :

1.Message Consists of a Premitive envelop some number of header fields a blank line and then the message body each header field consists of a single line of ASCII text containing the field name,a colon,and most field a value .

MIME-The Multipurpose Internet Mail Extension:

In the early days of the ARPANET,email consisted of text messages written in English and expressed in ASCII.for this environment, RFC 822 did the job completely:it specified the headers but left the content entirely up to the users.nowadays, on the world wide Internet ,this approach is no longer adequate.
The problems include sending and receiving

1. Message in languages with accents
2. Message in non-latin alphabets
3. Message in languages without alphabets
4. Message not containing text at all

A solution to the above problem is MIME.

MIME uses the RFC 822 format but to add structure to the message body and define encoding rules for non-ASCII messages.

2.4 Message Transfer

The message transfer system is concerned with relaying messages from the originator to the recipient.The simplest way to do this is to establish a transport connection from the source machine to the destination machine and then just transfer the message.

SMTP –THE SIMPLE MAIL TRANSFER PROTOCAL :

SMTP daemon accepts incoming connections and copies messaes from them into the appropriate mailboxes.if a message cannot be delivered an error report containing the first part of the undeliverable message is returned to the sender.

It is simple ASCII PROTOCAL

2.5 Final Delivery

POP3 –POST OFFICE PROTOCOL

Pop3 begins when the user starts the mail reader.The mail readers calls up the ISP and establishes a TCP connection with the message transfer agent at port 110.Once the connection has been Established,the pop3 protocol goes through three states in sequence.

- ✓ Authorization. –deals with user login.
- ✓ Transaction - deals with the user collecting the emails and making then for deletion from the mailbox.
- ✓ Update -this actually causes the e-mails to be deleted
- ✓

IMAP- INTERNET MESSAGE ACCESS PROTOCOL.

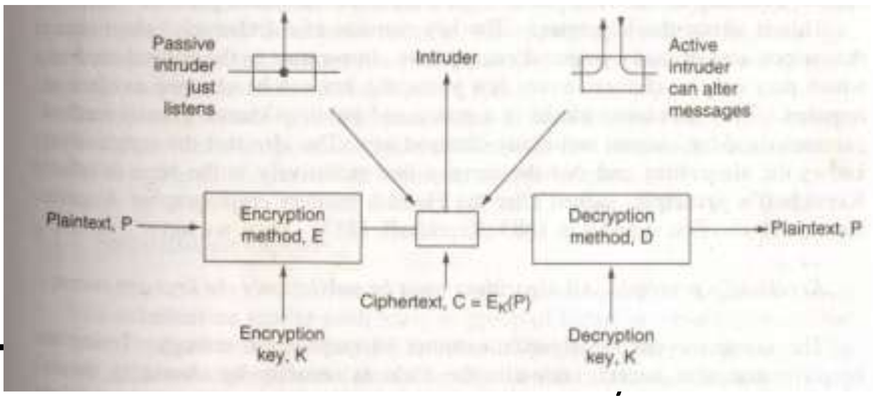
1. It Is Similar to pop3 protocol.
2. IMAP provides extensive mechanisms for reading messages or even part of a multipart messages with large audio and video attachments.
3. It can also accept outgoing email for shipments to the destination as well as deliver incoming e-mail.

4. <u>NETWORK SECURITY:</u>

<u>CRYPTOGRAPHY:</u>

Cryptography comes from the Greek words for "secret writing." Professionals make a distinction between ciphers and codes. A ciphers character-for-character or bit-for-bit transformation, without regard to the linguistic structure of the message. A code replaces one word with another word or symbol. Codes are not used any more.

<u>4.1 Introduction To Cryptography:</u>



Four groups of people have used and contributed to the art of cryptography. The military, the diplomatic corps, diarists, and lovers. The military has had the most important role and has shaped the field over the centuries. Within military organizations, the messages to be encrypted have traditionally been given to poorly-paid, low-level code clerks for encryption and transmission. The sheer volume of messages prevented this work from being done by a few elite specialists. Until the advent of computers, one of the main constraints on cryptography had been the ability of the code clerk to perform the necessary transformations, often on a battlefield with little equipment. An additional constraint has been the difficulty in switching over quickly from one cryptographic method to another one, since this entails retraining a large number of people. The danger of a code clerk being captured by the enemy has made it essential to be able to change the cryptographic method instantly if need be.

The Encryption Model (For A Symmetric-Key Cipher).

CIPHERTEXT

The messages to be encrypted, known as the plaintext, are transformed by a function that is parameterized by a **key**. The output of the encryption process, known as the **ciphertext**, is then transmitted, often by messenger or radio.

INTRUDER

We assume that the enemy or **intruder** hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know what the decryption key is and so cannot decrypt the ciphertext easily. Sometimes the intruder can not only listen to the communication channel (passive intruder) but can also record messages and play them back later, inject his own messages, or modify legitimate messages before they get to the receiver (active intruder). The art of breaking ciphers, called cryptanalysis and the art devising them (cryptography) is collectively known as Cryptology. It will often be useful to have a notation for relating plaintext, ciphertext and keys. We will use $C = E_K(P)$ to mean that the encryption of the plaintext key K gives the ciphertext C. Similarly, $P = D_K(C)$ represents the decryption of C to get the plaintext again. It then follows that $D_K(E_K(P)) = P$ .This notation suggests that E and D are just mathematical functions, which they are. The only tricky part is that both are functions of two parameters, and we have written one of the parameters (the key) as a subscript, rather than as an argument, to distinguish it from the message. A fundamental rule of cryptography is that one must assume that the cryptanalyst knows the methods used for encryption and decryption. In other words, the cryptanalyst knows how the encryption method, *E,* and decryption, *D.* The amount of effort necessary to invent, test, and install a new algorithm every time the old method is compromised (or thought to be compromised) has always made it impractical to keep the encryption algorithm secret.. The key consists of a (relatively) short string that selects one of many potential encryptions. In contrast to the general method, which may only be changed every few years, the key can be changed as often as required. Thus, our basic model is a stable and publicly-known general method parameterized by a secret and easily changed key. The idea that the cryptanalyst knows the algorithms and that the secrecy lies exclusively in the keys is called Kerckhoff's principle, named after the Flemish military cryptographer AuguSte Kerckhoff who first stated it in 1883 (Kerckhoff, 1883). Thus, we have:

Kerckhoff's principle:

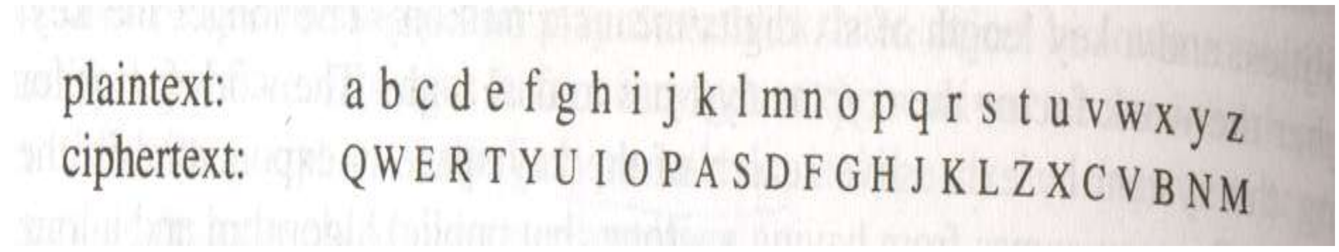
All algorithms must be public; only the keys are secret. The longer the key, higher the **work factor the** cryptanalyst has to deal with. •The work factor for breaking the system by exhaustive search of the key space is exponential in the key length. Secrecy comes from having a strong (but public) algorithm and a long key. To prevent your kid brother from reading your e-mail, 64-bit keys will do. For routine commercial use, at least 128 bits should be used. To keep major governments at bay. keys of at least *256* bits, preferably more, are needed. From the cryptanalyst's point of view, the cryptanalysis- problem has three principal variations. When he has a quantity of ciphertext and no plaintext, he is confronted with the **ciphertext-only** problem.

The cryptograms that appear in the puzzle section of newspapers pose this kind of problem. When the cryptanalyst has some matched ciphertext and plaintext, the problem is called the **known** plaintext problem. Finally, when the

cryptanalyst has the ability to encrypt pieces of plaintext of his own choosing, we have the **chosen plaintext problem.** Novices in the cryptography business often assume that if a cipher can withstand a ciphertext-only attack, it is secure. This assumption is very naive. In many cases the cryptanalyst can make a good guess at parts of the plaintext. Equipped with some matched plaintext-ciphertext pairs, the cryptanalyst's job becomes much easier. To achieve security, the cryptographer should be conservative and make sure that the system is unbreakable even if his opponent can encrypt arbitrary amounts of chosen plaintext. Encryption methods have historically been divided into two categories: substitution ciphers and transposition ciphers.

4.2 Substitution Ciphers:

In a **substitution cipher** each letter or group of letters is replaced by another letter or group of letters to disguise it. One of the oldest known ciphers is the attributed to Julius Caesar. In this method, a becomes *D, b* becomes *E, c* becomes *F,...,* and z becomes *C.* For example, *attack* becomes *WWDFN.* **In** examples, plaintext will be given in lower case letters, and cipher- ext in upper case letters. The next improvement is to have each of the symbols in the plaintext 26 letters for simplicity, map onto some other letter. For example, the general system of symbol-for-symbol substitution is called a monoalphabetic substitution, with the key being the 26-letter string corresponding to



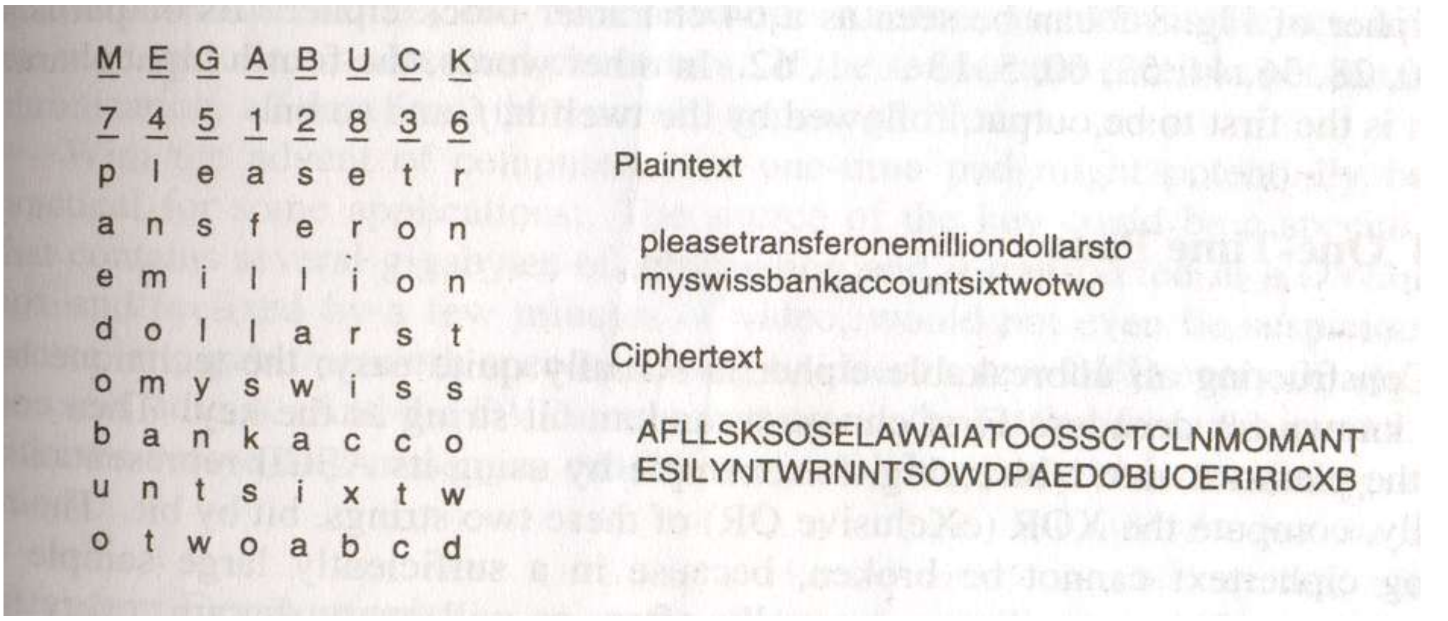
plaintext:	a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:	Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

the f ull alphabet. For the key above, the plaintext *attack* would be transformed into the ciphertext *QZZQEA.*
At first glance this might appear to be a safe system because although the cryptanalyst knows the general system (letter-for-letter substitution), he does not know which of the 26! 4 x 1026 possible keys is in use. Given a surprisingly small amount of ciphertext, the cipher can be broken easily. The

basic attack takes advantage of the statistical properties of natural languages. In English, for example, *e* is the most common letter, followed by *t,* o, a, *n, i,* etc. The most common two-letter combinations, or **digrams**, are *th, in, er, re,* and *an.* The most common three-letter combinations, or **trigrams**, are *the, ing, and,* and *ion.* A cryptanalyst trying to break a monoalphabetic cipher would start out by counting the relative frequencies of all letters in the ciphertext.

4.3 Transposition Ciphers:

Substitution ciphers preserve the order of the plaintext symbols but disguise them. Transposition ciphers, in contrast, reorder the letters but do not disguise them. Depicts a common transposition cipher, the columnar transposition. The cipher is keyed by a word or phrase not containing any repeated letters. This example, MEGABUCK is the key. The purpose of the key is to number the columns, column 1 being under the key letter closest to the start of the alphabet and so on. The plaintext is written horizontally, in rows, padded to fill the matrix if need be. The ciphertext is read out by columns, starting with the column whose key letter is the lowest.



M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Plaintext

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Ciphertext

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

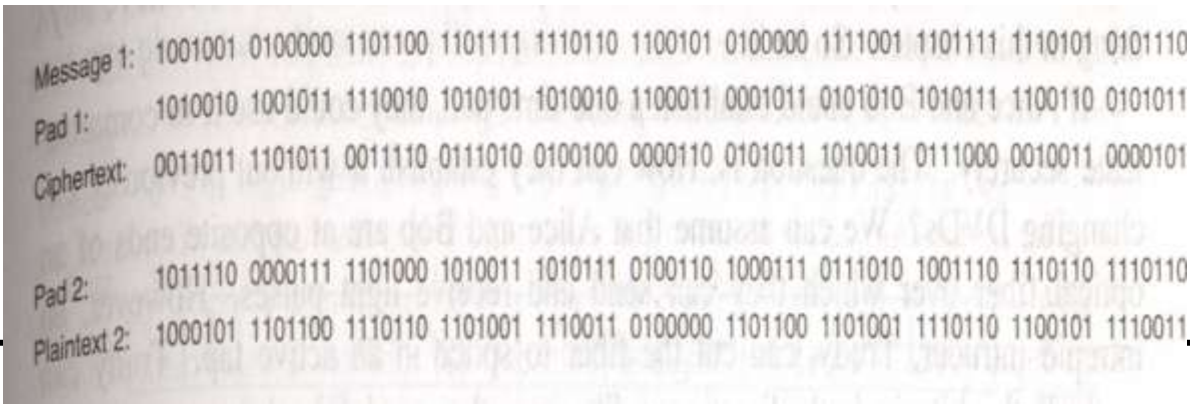
A transposition cipher.

To break a transposition cipher, the cryptanalyst must first be aware that he is dealing with a transposition cipher. By looking at the frequency of *E, T, A, 0, 1, N,* etc., t is easy to see if they fit the normal pattern for plaintext. The next step is to make a guess at the number of columns. In many cases a probable word or phrase may be guessed at from the context. For example, suppose that our cryptanalyst suspects that the plaintext phrase *milliondollars* occurs somewhere in the message. Observe that digrams *MO, IL, IL, LA, JR* and *OS* Occur in the ciphertext as a result of this phrase wrapping around. The ciphertext letter *0* follows the ciphertext letter *M* (i.e., they are vertically adjacent in column 4) because they are separated in the probable phrase by a distance equal to the key length. If a key of length seven had been used, the digrams *MD, 10, LL, LL, OR,* and *NS* would have occurred instead. In fact, for each key length, a different set of digrams is produced in the ciphertext. By hunting for the Various possibilities, the cryptanalyst can often easily determine the key length. The remaining step is to order the columns. When the number of columns, *k,* is small, each of the *k(k* - 1) column pairs can be examined to see if its digram frequencies match those for English plaintext. The pair with the best match is assumed to be correctly positioned. Now each remaining column is tentatively tried as the successor to this pair. The column whose digram and trigram frequencies give the best match is tentatively assumed to be correct. The predecessor column is found in the same way. The entire process is continued until a potential ordering is found. Chances are that the plaintext will be recognizable at this point (e.g., if *million* occurs, it is clear what the error is). Some transposition ciphers accept a fixed-length block of input and produce a fixed-length block of output. These ciphers can be completely described by giving a list telling the order in which the characters are to be output.

4.4 One-Time Pads:

First choose a random bit string as the key. Then convert the plaintext into a bit string, for example by using its ASCII representation. Finally, compute the XOR (exclusive OR) of these two strings, bit by bit. The resulting ciphertext cannot be broken, because in a sufficiently large sample of ciphertext, each letter will occur equally, as will every digram, every trigram, and so on. This method, known as the one-time pad, is immune to all present and future attacks no matter how much computational power the intruder has. The reason derives from information theory: there is simply no information in the message because all possible plaintexts of the given length are equally likely.

An example of how one-time pads are used is given in Fig. 8-4. First, message 1, "I love you." is converted to 7-bit ASCII. Then a one-time pad, pad 1 is chosen and XORed with the message to get the ciphertext. A cryptanalyst could try all possible one-time pads
see what plaintext came out for each one. For example, the one-time pad listed as pad 2 in the figure could be tried, resulting in plaintext 2, "Elvis lives", which may or may not be plausible. For every 11-character ASCII plaintext 1 there is a time pad that generates it. That is what we mean by saying there is no information in the ciphertext: you can get any message of the correct length out of it.



```
Message 1:   1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Pad 1:       1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Ciphertext:  0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Pad 2:       1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Plaintext 2: 1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011
```

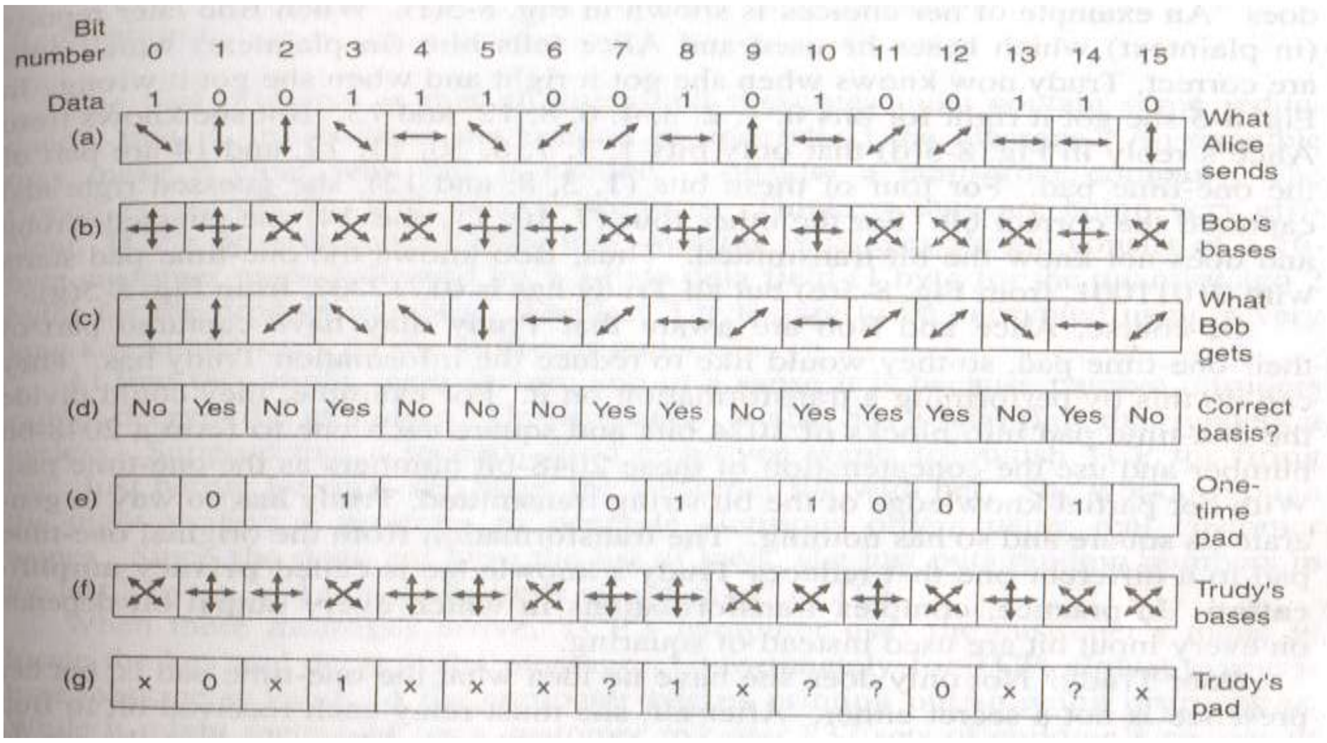
The use of a one-time pad for encryption and the possibility of getting any possible plaintext from the ciphertext by the use of some other pad.

One-time pads are great in theory but have a number of disadvantages in practice. To start with, the key cannot be memorized, so both sender and receiver must carry a written copy with them. If either
one is subject to capture, written keys are clearly undesirable. Additionally, the total amount of data that can be transmitted is limited by the amount of key available. If the spy strikes it rich and discovers a wealth of data, he may find himself unable to transmit it back to headquarters because the key has been used up. Another problem is the sensitivity of the method to lost or inserted characters. If the sender and receiver get out of synchronization, all data from then on will appear garbled.
With the advent of computers, the one-time pad might potentially become practical for some applications. The source of the key could be a special DVD that contains several gigabytes of information and if transported in a DVD movie box and prefixed by a few minutes of video, would not even be suspicious. Of course, at gigabit network speeds, having to insert a new DVD every 30 sec could become tedious. And the DVDs must be personally carried from the sender to the receiver before any messages can be sent, which greatly reduces their practical utility.

Quantum Cryptography:

There may be a solution to the problem of how to transmit the one-time pad over the network, and it comes from a very unlikely source: quantum mechanics. This area is still experimental, but initial tests are promising. If it can be perfected and be made efficient, virtually all cryptography will eventually be done using one-time pads since they are provably secure. We will describe a protocol called BB84 after its authors and publication year (Bennet and Brassard, 1984).
A user, Alice, wants to establish a one-time pad with a second user, Bob. Alice and Bob are called **principals**, the main characters in our story. For example, Bob is a banker with whom Alice would like to do business. If Alice and Bob could establish a one-time pad, they could use it to communicate securely. The question is: How can they establish it without previously exchanging DVDs? We can assume that Alice and Bob are at opposite ends of optical fiber over which they can send and receive light pulses. However an intrepid intruder, Trudy, can cut the fiber to splice in an active tap. Trudy

ca read all the bits in both directions. She can also send false messages in both directions. The situation might seem hopeless for Alice and Bob, but quantum cryptography can shed some new light on the subject. Quantum cryptography is based on the fact that light comes in little packets called **photons,** which have some peculiar properties. Furthermore, light can be polarized by being passed through a polarizing filter, a fact well known to both sunglasses wearers and photographers. If a beam of light (i.e., a stream of photons) is passed through a polarizing filter, all the photons emerging from it will be polarized in the direction of the filter's axis (e.g., vertical). If the beam is now passed through a second polarizing filter, the intensity of the light emerging from the second filter is proportional to the square of the cosine of the angle between the axes. If the two axes are perpendicular, no photons get through. The absolute orientation of the two filters does not matter; only the angle between their axes counts. To generate a one-time pad, Alice needs two sets of polarizing filters. Set one consists of a vertical filter and a horizontal filter. This choice is called a **rectilinear basis**. A basis (plural: bases) is just a coordinate system. The second set of filters is the same, except rotated *45* degrees, so one filter runs from the lower left to the upper right and the other filter runs from the upper left to the lower right. This choice is called a **diagonal basis**.



Bit number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Data	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	
(a)	⤡	↕	↕	⤡	↔	⤡	⤢	⤢	↔	↕	↔	⤢	⤢	↔	↔	↕	What Alice sends
(b)	✢	✢	✕	✕	✕	✢	✢	✕	✢	✕	✢	✕	✕	✕	✢	✕	Bob's bases
(c)	↕	↕	⤢	⤡	⤡	↕	↔	⤢	↔	⤢	↔	⤢	⤢	⤡	↔	⤢	What Bob gets
(d)	No	Yes	No	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No	Correct basis?
(e)		0		1				0	1		1	0	0		1		One-time pad
(f)	✕	✢	✢	✕	✢	✢	✕	✢	✢	✕	✕	✢	✕	✢	✕	✕	Trudy's bases
(g)	×	0	×	1	×	×	×	?	1	×	?	?	0	×	?	×	Trudy's pad

Thus, Alice has two bases, which she can rapidly insert into her beam at will. In reality, Alice does not have four separate filters, but a crystal whose polarization can be switched electrically to any of the four allowed directions at great speed. Bob has the same equipment as Alice. The fact that Alice and Bob each have two bases available is essential to quantum cryptography. For each basis, Alice now assigns one direction as 0 and the other as 1. In the example presented below, we assume she chooses vertical to be 0 and horizontal to be 1. Independently, she also chooses lower left to upper right as 0 and upper left to lower right as 1. She sends these choices to Bob as plaintext. Now Alice picks a one-time pad, for example based on a random number generator (a complex subject all by itself). She transfers it bit by bit to Bob, choosing its one of her two bases at random for each bit. To send a bit, her photon gun - one photon polarized appropriately for the basis she is using for that bit. She might choose bases of diagonal, rectilinear, rectilinear, diagonal, and rectilinear etc. To send her one-time pad of 1001110010100110 with these bases, she would send the photons shown in Fig. Given the one-time pad and the sequence of bases, the polarization to use for each bit is uniquely determined. Bits sent one photon at a time are called **qubits.** Bob does not know which bases to use, so he picks one at random for each arriving photon and just uses it. If he picks the correct basis, he gets the correct bit. If he picks the incorrect basis, he gets a random bit because if a photon hits a filter polarized at *45* degrees to its own polarization, it randomly jumps to the polarization of the filter or to a polarization perpendicular to the filter with equal probability. This property of photons is fundamental to quantum mechanics. Thus, some of the bits are correct and some are random, but bob does not know which are which. Bob's results are depicted in Fig. How does Bob find out which bases he got right and which he got wrong? He simply tells Alice which basis he used for each bit in plaintext and she tells him which this are right and which are wrong in plaintext as shown in Fig. From this information both of them can build a bit string from the correct guesses, as in Fig. On the average, this bit string will be half the length of the original bit String, but since both parties know it, they can use it as a one-time pad. All alice has to do is transmit a bit string slightly more than twice the desired length she and Bob have a one-time pad of the desired length. Problem solved. But wait a minute. We forgot Trudy. Suppose that she is curious about what Alice has to say and cuts the fiber, inserting her own detector and transmitter.

Unfortunately for her, she does not know which basis to use for each photon either. The best she can do is pick one at random for each photon, just as Bob does. An example of her choices is shown in Fig. When Bob later reports (in plaintext) which bases he used and Alice tells him (in plaintext) which ones are correct, Trudy now knows when she got it right and when she got it wrong Fig. she got it right for bits 0, 1, 2, 3, 4, 6, 8, 12, and 13. But she knows from Alice's reply in Fig that only bits 1, 3, 7, 8, 10, 11, 12, and 14 are part of their one-time pad. For four of these bits (1, 3, 8, and 12), she guessed right and captured the correct bit. For the other four (7, 10, 11, and 14) she guessed wrong and does not know the bit transmitted. Thus, Bob knows the one-time pad starts with 01011001, from Fig but all Trudy has is 01?1??0?, from Fig.
Of course, Alice and Bob are aware that Trudy may have captured part of their one-time pad, so they would like to reduce the information Trudy has. They can do this by performing a transformation on it. For example, they could divide the one-time pad into blocks of 1024 bits and square each one to form a 2048-bit number and use the concatenation of these 2048-bit numbers as the one-time pad. With her partial knowledge of the bit string transmitted, Trudy has no way to generate its square and so has nothing. The transformation from the original one-time pad to a different one that reduces Trudy's knowledge is called **privacy amplification**. In practice, complex transformations in which every output bit depends on every input bit are used instead of squaring.

4.5 Two Fundamental Cryptographic Principles:

Redundancy:

Cryptographic principle 1: Messages must contain some redundancy

Freshness:

Cryptographic principle 2: Some method is needed to foil replay attacks

5. SYMMETRIC-KEY ALGORITHMS:

Modern cryptography uses the same basic ideas as traditional cryptography (transposition and substitution) but its emphasis is different. Traditionally, cryptographers have used simple algorithms. The first classes of encryption algorithms are called symmetric-key algorithms because they used the same key for encryption and decryption. **Block ciphers,** which take an n-bit block of plaintext as input and transform it using the key into n-bit block of ciphertext.

Cryptographic algorithms can be implemented in either hardware (for speed) or in software (for flexibility). Although most of our treatment concerns the algorithms and protocols, which are independent of the actual implementation, a few words about building cryptographic hardware may

be of interest. Transpositions and substitutions can be implemented with simple electrical circuits. Figure 8- 6(a) shows a device, known as a **P-box** (P stands for permutation), used to effect a transposition on an 8-bit input. If the 8 bits are designated from top to bottom as *01234567,* the output of this particular P-box is *36071245.* By appropriate internal wiring, a P-box can be made to perform any transposition and do it at practically the speed of light since no computation is involved, just signal propagation. This design follows Kerckhoff's principle: the attacker knows that the general method is permuting the bits. What he does not know is which bit goes where, which is the key.

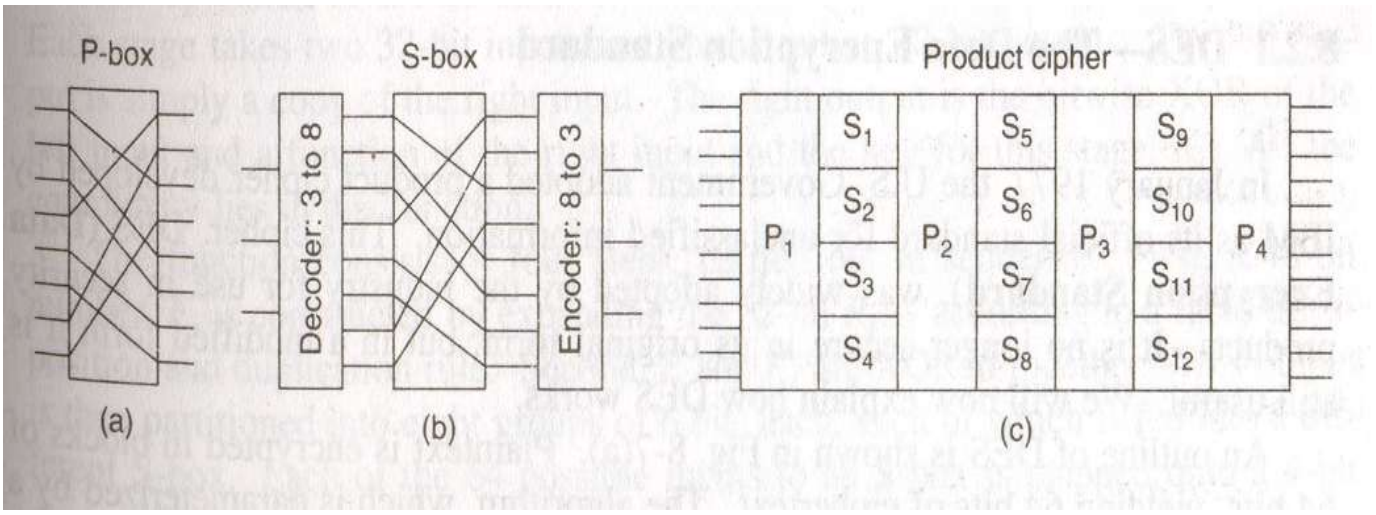


Figure 8-6. Basic elements of product ciphers. (a) P-box. (b) S-box. (c) Product.

Substitutions are performed by S-boxes, as shown in Fig. 8-6(b). In this example a 3-bit plaintext is entered and a 3-bit ciphertext is output. The 3-bit input selects one of the eight lines exiting from the first stage and sets it to 1; all the Other lines are 0. The second stage is a P-box. The third stage encodes the selected input line in binary again. With the wiring shown, if the eight octal numbers 01234567 were input one after another, the output sequence would be 24506713. In other words, 0 has been replaced by 2, 1 has been replaced by *4,* etc. Again by appropriate wiring of the P-box inside the

S-box, any substitution can be accomplished. Such a device can be built in hardware and can achieve great speed since encoders and decoders have only one or two (subnanosecond) gate delays and the propagation time across the P-box may well be less than 1 picosecond.

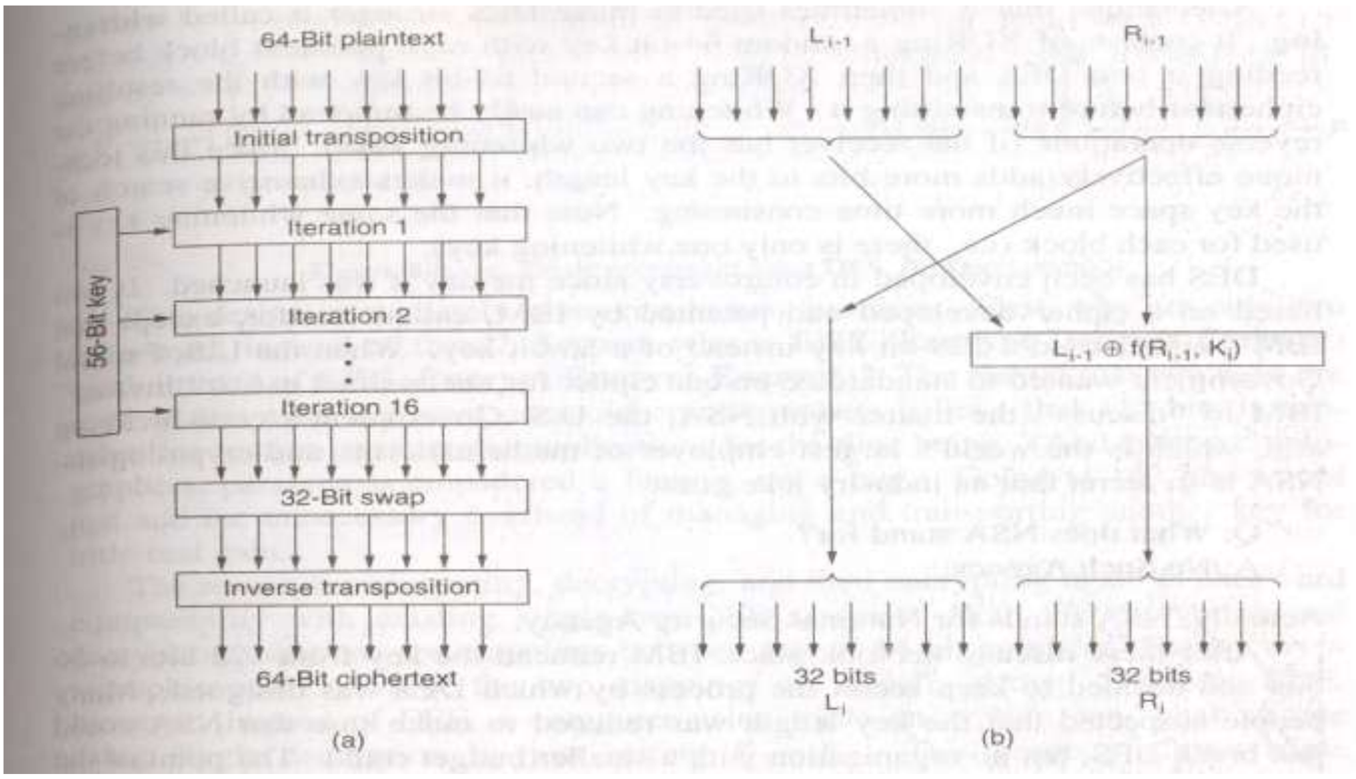
The real power of these basic elements only becomes apparent when we cascade a whole series of boxes to form a **product cipher,** as shown in Fig 86(c) In this example, 12 input lines are transposed (i.e., permuted) by the first stage *(P_1).* Theoretically, it would be possible to have the second stage be an S-box that mapped a 12-bit number onto another 12-bit number. However, such a device would need 2^{12} = 4096 crossed wires in its middle stage. Instead, the input is broken up into four groups of 3 bits, each of which is substituted independently of the others. Product ciphers that operate on k-bit inputs to produce k-bit outputs are very common. Typically, k is 64 to 256. A hardware implementation usually has at least 18 physical stages, instead of just seven as in Fig. 8-6(c). A software implementation is programmed as a loop with at least 8 iterations, each one performing S-box-type substitutions on subblocks of the 64- to 256-bit data block, followed by a permutation that mixes the outputs of the S-boxes. Often there is a special initial permutation and one at the end as well. In the literature, the iterations are called **rounds.**

5.1 DES—THE DATA ENCRYPTION STANDARD:

In January 1977, the U.S. Government adopted a product cipher developed by IBM as its official standard for unclassified information. This cipher, DES (Data **Encryption** Standard), was widely adopted by the industry for use in security products. It is no longer secure in its original form, but in a modified form it is still useful. We will now explain how DES works.

An outline of DES is shown in Fig. 8-7(a). Plaintext is encrypted in blocks of 64 bits, yielding 64 bits of ciphertext. The algorithm, which is parameterized by a 56-bit key, has 19 distinct stages. The first stage is a key-independent transposition on the 64-bit plaintext. The last stage is the exact inverse of this transposition. The stage prior to the last one exchanges the leftmost 32 bits with the rightmost 32 bits. The remaining 16 stages are functionally

identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption, a property need in any symmetric-key algorithm. The steps are just run in the reverse order.



The data encryption standard. (a) General outline. (b) Detail of one iteration. The circled + means exclusive OR.

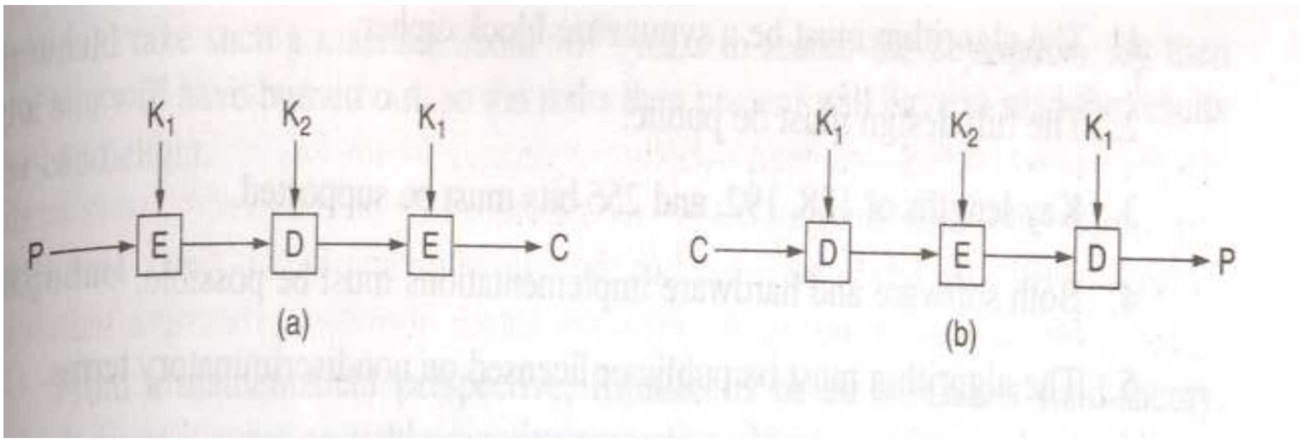
The operation of one of these intermediate stages is illustrated in Fig. 8-7(b). Each stage takes two 32-bit inputs and produces two 32-bit outputs. The left output is simply a copy of the right input. The right output is the bitwise XOR of the left input and a function of the right input and the key for this stage, K_i. All the complexity lies in this function. The function consists of four steps, carried out in sequence. First, a 48-bit number, *E,* is constructed by expanding the 32-bit R_{i-1} according to a fixed transposition and duplication rule. Second, *E* and K_i are XORed together. This output is then partitioned into eight groups of 6 bits each, each of which is fed into a different S-box. Each of the 64 possible inputs to an S-box is mapped onto a 4-bit Output. Finally, these 8 x 4 bits are passed through a P-box. In each of the 16 iterations, a different key is used. Before the algorithm starts, a 56-bit transposition is applied to the key. Just before each iteration, the key is partitioned into two 28-bit units, each of which is rotated left by a number of bits dependent on the iteration number. K_i is derived from this rotated key by applying yet another 56-bit transposition to **it.** A different 48-bit subset of the 56 bits is extracted and permuted on each round.

A technique that is sometimes used to make DES stronger is called whitening. It consists of XORing a random 64-bit key with each plaintext block before feeding it into DES and then XORing a second 64-bit key with the resulting ciphertext before transmitting it. Whitening can easily be removed by running reverse operations (if the receiver has the two whitening keys). Since this technique effectively adds more bits to the key length, it makes exhaustive search of the key space much more time consuming. Note that the same whitening key is used for each block (i.e., there is only one whitening key). DES has been enveloped in controversy since the day it was launched It was based on a cipher developed and patented by IBM, called Lucifer, except that IBM's cipher used a 128-bit key instead of a 56-bit key. NSA stands for National Security Agency. IBM reduced the key from 128 bits to *56* bits and decided to keep secret the process by which DES was designed. Many people suspected that the key length was reduced to make sure that NSA could just break DES, but no organization with a smaller budget could. The point of the secret design was supposedly to hide a back door that could make it even easier for NSA to break DES. When an NSA employee discreetly told IEEE to cancel a planned conference on cryptography that did not make people any more comfortable. NSA denied everything. In 1977, two Stanford cryptography researchers, Diffie and Heliman (1977), designed a machine to break DES and estimated that it could be built for 20 million
dollars. Given a small piece of plaintext and matched ciphertext, this machine could find the key by exhaustive search of the 256-entry key space in under 1 day. Nowadays, such a machine would cost well under 1 million dollars.

TRIPLE DES:

As early as 1979, IBM realized that the DES key length was too short and7 vised a way to effectively increase it, using triple encryption (Tuchmandard The method chosen, which has since been incorporated in International In the 8732, is illustrated in Fig. 8-8. Here two keys and three stages are used. In the first stage, the plaintext is encrypted using DES in the usual way with Ki. anoth second stage, DES is run in decryption mode, using J2 as the key. Fina Y
er DES encryption is done with *K1.*

Triple encryption using DES. (b) Decryption.



This design immediately gives rise to two questions. First, why are only two keys used, instead of three? Second, why is EDE (Encrypt Decrypt Encrypt) used, instead of EEE (Encrypt Encrypt Encrypt)? The reason that two keys are used is that even the most paranoid cryptographers believe that 112 bits is adequate for routine commercial applications for the time being. (And among cryptographers, paranoia is considered a feature, not a bug.) Going to 168 bits would just add the unnecessary overhead of managing and transporting another key for little real gain.
The reason for encrypting, decrypting, and then encrypting again is backward compatibility with existing single-key DES systems. Both the encryption and decryption functions are mappings between sets of 64-bit numbers. From a cryptographic point of view, the two mappings are equally strong. By using EDE, however, instead of EEE, a computer using triple encryption can speak to one using single encryption by just setting *K1 = K2.* This property allows triple encryption to be phased in gradually, something of no concern to academic cryptographers, but of considerable importance to IBM and its customers.

5.2 AES—THE ADVANCED ENCRYPTION STANDARD:

As DES began approaching the end of its useful life, even with triple DES, NIST (National Institute of Standards and Technology), the agency of the U.S. Dept. of Commerce charged with approving standards for the U.S. Federal Government, decided that the government needed a new cryptographic standard for Unclassified use. NIST was keenly aware of all the controversy surrounding DES and well knew that if it just

announced a new standard, everyone knowing anything about cryptography would automatically assume that NSA had built a back door into it so NSA could read everything encrypted with it. Under these conditiOflS, probably no one would use the standard and it would most likely die a quiet death
So NIST took a surprisingly different approach for a government bureaucracy: **it** Sponsored a cryptographic bake-off (contest). In January 1997, researchers from all over the world were invited to submit proposals for a new standard, to be called AES (Advanced Encryption Standard). The bake-off rules were:

6. <u>PUBLIC-KEY ALGORITHMS:</u>

Historically, distributing the keys has always been the weakest link in cryptosystems. No matter how strong a cryptosystem was, if an intruder c5 steal the key, the system. was worthless. Cryptologists always took for gran that the encryption key and decryption key were the same (or easily derived from one another). But the key had to be distributed to all users of the system. Thus seemed as if there was an inherent built-in problem. Keys had to be protected from theft,

but they also had to be distributed, so they could not just be locked u in a bank vault. p
In 1976, two researchers at Stanford University, Diffie and Hellman (1976) proposed a radically new kind of cryptosystem, one in which the encryption anci decryption keys were different, and the decryption key could not feasibly be derived from the encryption key. In their proposal, the (keyed) encryption algorithm, *E,* and the (keyed) decryption algorithm, *D,* had to meet three requirements. These requirements can be stated simply as follows:

1. $D(E(P))=P$.
2. It is exceedingly difficult to deduce D from E.
3. E cannot be broken by a chosen plaintext attack.

The first requirement says that if we apply D to an encrypted message, $E(P)$, we get the original plaintext message, P, back. Without this property, the legitimate receiver could not decrypt the ciphertext. The second requirement speaks for itself. The third requirement is needed because, as we shall see in a moment, intruders may experiment with the algorithm to their hearts' content. Under these conditions, there is no reason that the encryption key cannot be made public.

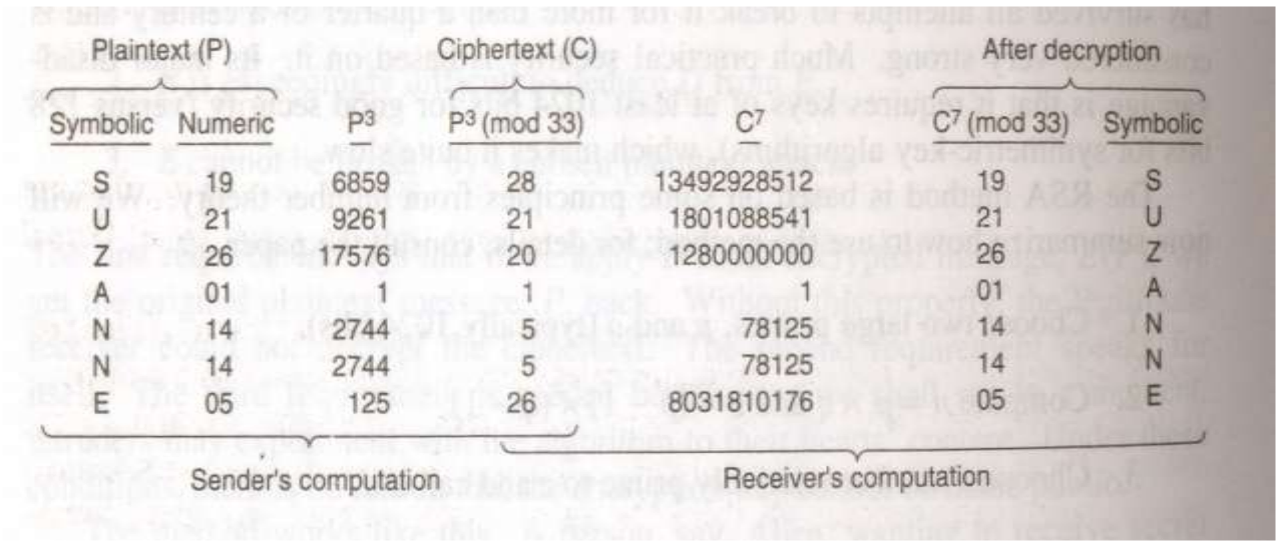
The method works like this. A person, say, Alice, wanting to receive secret messages, first devises two algorithms meeting the above requirements. The encryption algorithm and Alice's key are then made public, hence the name public key cryptography. Alice might put her public key on her home page on the Web, for example. We will use the notation *EA* to mean the encryption algorithm parameterized by Alice's public key. Similarly, the (secret) decryption algorithm parameterized by Alice's private key is ***DA.*** Bob does the same thing, publlclzing\ *EB* but keeping *DB* secret.
Now let us see if we can solve the problem of establishing a secure chanfle between Alice and Bob, who have never had any previous contact. Both Alice S encryption key, *EA,* and Bob's encryption key, *EB,* are assumed to be in publiClY readable files. Now Alice takes her first message, *P,* computes *E8(P),* and sen it to Bob. Bob then decrypts it by applying his secret key D_8 [i.e., he computeS *D8(E8(P)) =P].* No one else can read the encrypted message, *E8(P),* because the encryption system is assumed strong and because it is too difficult to derive *D* from the publicly known ***EB.*** **To send a reply,** *R,* Bob transmits *EA(R).* Alice and Bob can now communicate securely.
A note on terminology is perhaps useful here. Public-key cryptography reuires each user to have two keys: a public key, used by the entire world for enrypting messages to be sent to that user, and a private key, which the user needs for deciyptmg messages. We will consistently refer to these keys as the *public* and *private* keys, respectively, and distinguish them from the *secret* keys used for conventional symmetric-key cryptography.

6.1 RSA :

The only catch is that we need find algorithms that indeed satisfy all three requirements. Due to the potential advantages of public-key cryptography, many researchers are hard at work, and some algorithms have already been published. One good method was discovered by a group at M.I.T. (Rivest et al., 1978), It is known by the initials of the three discoverers (Rivest, Shamir, Adleman): **RSA. It** has survived all attempts to break it for more than a quarter of a century and is considered very strong. Much practical security is based on it. Its major disadvantage is that it requires keys of at least 1024 bits for good security (versus 128 bits for symmetric-key algorithms), which makes it quite slow,
The RSA method is based on some principles from number theory. We will now summarize how to use the method; for details, consult the paper.1. Choose two large

primes, *p* and q (typically 1024 bits).
2. Computen *—p xqandz=(p—l)x(q--* 1).
3. Choose a number relatively prime to z and call it *d.*
4. Find *e* such that *e x d* = 1 *mod* z.

With these parameters computed in advance, we are ready to begin encryption. Divide the plaintext (regarded as a bit string) into blocks, so that each plaintext message, *P.* falls in the interval 0 *P <n.* Do that by grouping the plaintext into blocks of *Ic* bits, where *k* is the largest integer for which ***2k*** *<n* is true. To encrypt a message, *P,* compute c = *P'* (mod *n).* To decrypt *C,* compute *P C'* (mod **17).** It can be proven that for all *P* in the specified range, the encryption and decryption functions are inverses. To perform the encryption, you need *e* and *n.* To perform the decryption, you need *d* and *n.* Therefore, the public key **COflS1StS** of the pair *(e, n),* and the private key consists *of(d, n).* The security of the method is based on the difficulty of factoring large num bers If the cryptanalyst could factor the (publicly known) *n,* he could then find p and q, and from these z. Equipped with knowledge of z and *e, d* can be found using Euclid's algorithm. Fortunately, mathematicians have been trying to f large numbers for at least 300 years, and the accumulated evidence Suggests is an exceedingly difficult problem. at **t** According to Rivest
and colleagues, factoring a 500-digit number requ1 1025 years using brute force. In both cases, they assume the best known algoritS and a computer with a I—l. t sec instruction time. Even if computers Continue to m faster by an order of magnitude per decade, it will be centuries before factorint 500-digit number becomes feasible, at which time our descendants can simply choose p and q still larger.
A trivial pedagogical example of how the RSA algorithm works is given in Fig. 8-17. For this example we have chosen p = 3 and q = 11, giving fl = 33 and z = 20. A suitable value for *d* is *d* = 7, since 7 and 20 have no common factors. With these choices, *e* can be found by solving the equation *7e* = I (mod 20), which yields *e* = 3. The ciphertext, *C,* for a plaintext message, *P,* is given by *C P3* (mod 33). The ciphertext is decrypted by the receiver by making use of the rule *P* = *C7* (mod 33). The figure shows the encryption of the plaintext "SUZANNE" as an example.



Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Sender's computation				Receiver's computation		

Figure 8-17. An example of tle RSA algorithm. Because the primes chosen for this example are so small, *P* must be less than 33, so each plaintext block can contain only a single character. The result is a monoalphabetic substitution cipher, not very impressive. If instead we had chosen p and q 2512, we would have *n* 21024, so each block could be up to 1024 bits or 128 eight-bit characters, versus 8 characters for DES and 16 character for AES. It should be pointed out that using RSA as we have described is similar to the same using a symmetric algorithm in ECB mode—the same input oioCis gives output block. Therefore, some form of chaining is needed for data enCrYPtbOT However, in practice, most RSA-based **systems** use public-key cryptOgraPt1Y manly for distributing one-time session keys for use with some symmetric algorithm such as AES or triple DES. RSA is too slow for actually encrypt large volumes of data but is widely used for key distribution.

<u>6.2 Other Public-Key Algorithms</u>:

Although RSA is widely used, it is by no means the only pubic-key algorithm . The first public-key algorithm was the knapsack algorithm (Merkie and IJellma1 1978). The idea here is that someone owns a large number of objects, each with a different weight. The owner encodes the message by secretly selecting a subset of the objects and placing them in the knapsack. The total weight of the objects in the knapsack is made public, as is the list of all possible objects. The list of objects in the knapsack is kept secret. With certain additional restrictions, the problem. of figuring out a possible list of objects with the given weight was thought to be computationally infeasible and formed the basis of the public- key algorithm. The algorithm's inventor, Ralph Merkie, was quite sure that this algorithm could not be broken, so he offered a $100 reward to anyone who could break it. Adi Shamir (the "S" in RSA) promptly broke it and collected the reward. Undeterred, Merkle strengthened the algorithm and offered a $1000 reward to anyone who could break the new one. Ronald Rivest (the "R" in RSA) promptly broke the new one and collected the reward. Merkie did not dare offer $10,000 for the next version, so "A" (Leonard Adleman) was out of luck. Nevertheless, the knapsack algorithm is not considered secure and is not used in practice any more.

Other public-key schemes are based on the difficulty of computing discrete logarithms. Algorithms that use this principle have been invented by El Gamal (1985) and Schnorr (1991).

A few other schemes exist, such as those based on elliptic curves (Menezes and Varistone, 1993), but the two major categories are those based on the difficulty of factoring large numbers and computing discrete logarithms modulo a large prime. These problems are thought to be genuinely difficult to solve—mathematicians have been working on them for many years without any great breakthroughs.

7. DIGITAL SIGNATURES:

The authenticity of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature. And photocopies do not count. For computerized message systems to replace the physical transport of paper and ink documents, a method must be found to allow documents to be signed in an unforgeable way.

The problem of devising a replacement for handwritten signatures is a difficult one. Basically, what is needed is a system by which one party can send a Signed message to another party in such a way that the following conditions hold:

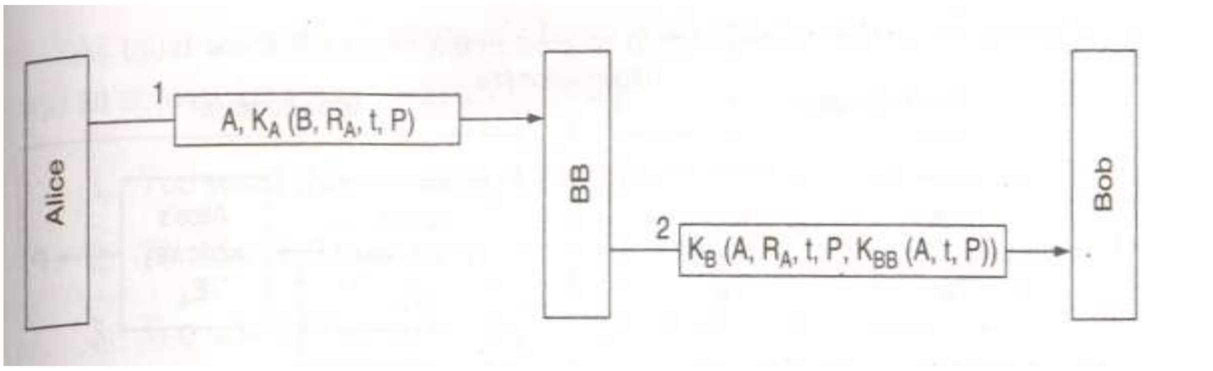
1. The receiver can verify the claimed identity of the sender.
2. The sender cannot later repudiate the contents of the message,
3. The receiver cannot possibly have concocted the message himself.

The first requirement is needed, for example, in financial systems When customer's computer orders a bank's computer to buy a ton of gold, the bank's computer needs to be able to make sure that the computer giving the order reall belongs to the company whose account is to be debited. In other words, the bank has to authenticate the customer (and the customer has to authenticate the bank). The second requirement is needed to protect the bank against fraud. Suppose that the bank buys the ton of gold, and immediately thereafter the price of gold drops sharply. A dishonest customer might sue the bank, claiming that he never issued any order to buy gold. When the bank produces the message in court, the customer denies having sent it. The property that no party to a contract can later deny having signed it is called **nonrepudiation.** The digital signature schemes that we will now study help provide it. The third requirement is needed to protect the customer in the event that the price of gold shoots up and the bank tries to construct a signed message in which the customer asked for one bar of gold instead of one ton. In this fraud scenario, the bank just keeps the rest of the gold for itself.

7.1 SYMMETRIC-KEY SIGNATURES:

One approach to digital signatures is to have a central authority that knows everything and whom everyone trusts, say Big Brother *(BB).* Each user then chooses a secret key and carries it by hand to *BB's* office. Thus, only Alice and *BB* know Alice's secret key, *KA,* and so on.

When Alice wants to send a signed plaintext message, *P,* to her banker, Bob, she generates *KA(B, RA, t, P),* where *B* is Bob's identity, *RA* is a random number chosen by Alice, *t* is a timestamp to ensure freshness, and *KA(B, RA, 1, P)* is the message encrypted with her key, *KA.* Then she sends it as depicted in Fig. 8-18. *BB* sees that the message is from Alice, decrypts it, and sends a message to Bob as shown. The message to Bob contains the plaintext of Alice's message and also the signed message *KBB (A, t, P).* Bob now carries out Alice's request. What happens if Alice later denies sending the message? Step I is that ever Y one sues everyone (at least, in the United States). Finally, when the case comes e court and Alice vigorously denies sending Bob the disputed message, the ju g will ask Bob how he can be sure that the disputed message came from **Al1Ce** not from Trudy. Bob first points out that *BB* will not accept a message from *BB* a unless it is encrypted with *KA,* so there is no possibility of Trudy sending false message from Alice without BB detecting it immediately.

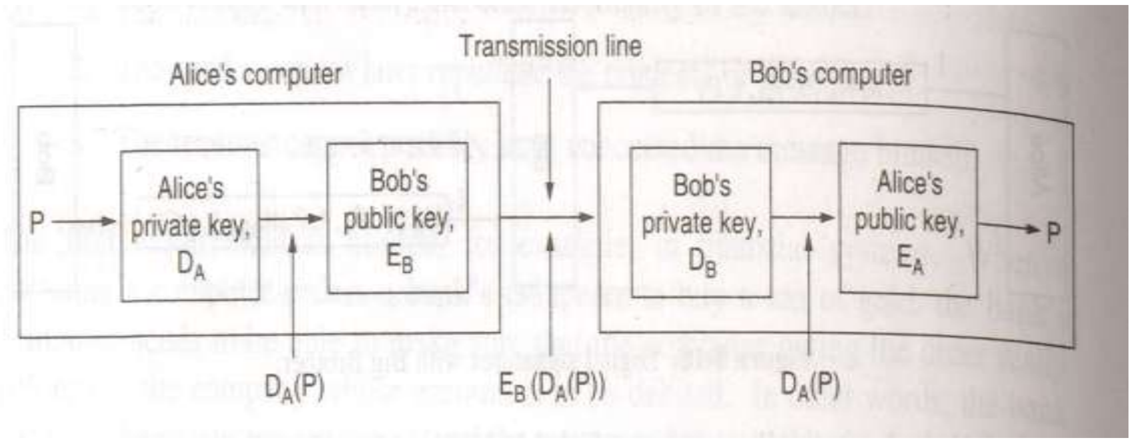

Alice
1
A, K_A (B, R_A, t, P)
BB
2
K_B (A, R_A, t, P, K_{BB} (A, t, P))
Bob

Digital signatures with Big Brother.

Bob then dramatically produces Exhibit A: *KBB(A,* **t,** *P).* Bob says that this is a message signed by *BB* which proves Alice sent *P* to Bob. The judge then asks *BB* (whom everyone trusts) to decrypt Exhibit A. When *BB* testifies that Bob is telling the truth, the judge decides in favor of Bob. Case dismissed.
One potential problem with the signature protocol of Fig. 8-18 is Trudy re playing either message. To minimize this problem, timestamps are used throughout. Furthermore, Bob can check all recent messages to see if *RA* was used in any of them. If so, the message is discarded as a replay. Note that based on the timestamp, Bob will reject very old messages. To guard against instant replay attacks, Bob just checks the *RA* of every incoming message to see if such a message has been received from Alice in the past hour. If not, Bob can safely assume this is a new request.

7.2 <u>PUBLIC-KEY SIGNATURES:</u>

A structural problem with using symmetric-key cryptography for digital signatures is that everyone has to agree to trust Big Brother. Furthermore, Big Brother gets to read all signed messages. The most logical candidates for running the Big Brother server are the government, the banks, the accountants, and the lawyers. Unfortunately, none of these organizations inspire total confidence in all citizens. Hence, it would be nice if signing documents did not require a trusted authority. Fortunately, public-key cryptography can make an important contribution in this area. Let us assume that the public-key encryption and decryption algorithms have the property that *E(D(P)) = P* in addition, of course, to the usual property that *D(E(P)) = P.* (RSA has this property, so the assumption is not unreasonable.) Assuming that this is the case, Alice can send a signed plaintext message, *P,* to Bob by transmitting *EB (DA(P)).* Note carefully that Alice knows her own (pnvate) key, ***DA,*** as well as Bob's public key, *EB,* so constructing this message is Something Alice can do. When Bob receives the message, he transforms it using his private key, as Usual, yielding *DA (P),* as shown in Fig. 8-19. He stores this text in a safe place and then applies *EA* to get the original plaintext.



Digital signatures using public-key cryptography.

To see how the signature property works, suppose that Alice subsequently denies having sent the message *P* to Bob. When the case comes up in court, Bob can produce both *P* and *DA(P).* The judge can easily verify that Bob indeed has a valid message encrypted by *DA* by simply applying *EA* to it. Since Bob does not know what Alice's private key is, the only way Bob could have acquired a message encrypted by it is if Alice did indeed send it. While in jail for perjury and fraud. Alice will have plenty of time to devise interesting new public-key algorithms.

Although using public-key cryptography for digital signatures is an elegant scheme, there are problems that are related to the environment in which they operate rather than with the basic algorithm. For one thing, Bob can prove that a message was sent by Alice only as long as *DA* remains secret. If Alice discloses her secret key, the argument no longer holds, because anyone could have sent the message, including Bob himself. The problem might arise, for example, if Bob is Alice's stockbroker. Alice tells Bob to buy a certain stock or bond. Immediately thereafter, the price drops sharply. To repudiate her message to Bob, Alice runs to the police claiming that her home was burglarized and the PC holding her key was stolen. Depending **O** the laws in her state or country, she may or may not be legally liable, especially **If** she claims not to have discovered the break-in until getting home from work, several hours later. Another problem with the signature scheme is what happens if Alice decides to change her key. Doing so is clearly legal, and it is probably a good idea to do so periodically. If a court case later arises, as described above, the judge **Wil** apply the *current EA* to ***DA(P)*** and discover that it does not produce *P.* Bob **WI** look pretty stupid at this point. The In principle, any public-key algorithm can be used for digital signatures. de facto industry standard is the RSA algorithm. Many security products use ' However, in 1991, NIST proposed using a variant of the El Gamal bliC-keY gorithm for their new **Digital Signature Standard (DSS).** El Gamal gets **te** security from the difficulty of computing discrete logarithms, rather than from difficulty of factoring large numbers. As usual when the government tries to dictate cryptographic standards, there was an uproar. DSS was criticized for being

I. Too secret (NSA designed the protocol for using El Gamal).
2. Too slow (10 to 40 times slower than RSA for checking signatures).
3. Too new (El Gamal had not yet been thoroughly analyzed).
4. Too insecure (fixed 512-bit key).

In a subsequent revision, the fourth point was rendered moot when keys up to 1024 bits were allowed. Nevertheless, the first two points remain valid.

MODEL QUESTIONS:

1. Explain DNS in Detail?
2. Write about User Agents?
3. Describe Network Security?
4. Write short note on Email-Architecture and Services?