ANNAMACHARYA
INSTITUTE OF TECHNOLOGY AND SCIENCES
(AUTONOMOUS)

Approved by AICTE, New Delhi & Permanent Affiliation to JNTUA, Anantapur.

Three B. Tech Programmes (CSE , ECE & CE) are accredited by NBA, New Delhi,Accredited by NAAC with 'A' Grade , Bangalore.

A-grade awarded by AP Knowledge Mission. Recognized under sections 2(f) & 12(B) of UGC Act 1956.

Venkatapuram Village, Renigunta Mandal, Tirupati, Andhra Pradesh-517520.

Department of Artificial Intelligence

Academic Year 2023-24

III. B.Tech I Semester

Computer Networks & Cryptography

(20APC3315)

Prepared By

Mrs. D.Divyashree.,

M.Tech(Ph.D),

Assistant Professor

Department of AI, AITS

ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES::TIRUPATI (AUTONOMOUS)

Year: III **Semester: I** **Branch of Study: AI & ML**

COURSE CODE	COURSE TITLE	L	T	P	CREDITS
20APC3315	Computer Networks & Cryptography	3	0	0	3

Course Outcomes:

After Completion of this course, the student will be able to:

CO1: Understand the basics of data communications and networking

CO2: Classify the functionalities of two sub layers of Data link Layer

CO3: Know briefly about Network Layer through algorithms and protocols

CO4: Distinguish the services provided by Transport Layer

CO5: Recognize the services offered by Application Layer to the user

Unit -1:

Introduction: Data Communications, Networks, Network Types, Internet History, Standards andAdministration.
Network Models: Protocol Layering, TCP/IP Protocol Suite, The OSI Model
Introduction to Physical Layer: Data and Signals, Transmission Impairment, Data Rate Limits,Performance.
Transmission Media: Introduction, Guided Media, Unguided Media.

Unit -2:

The Data Link Layer: Introduction, Link layer addressing, Error detection and Correction: Cyclic codes, Checksum, Forward error correction, Data link control: DLC Services, Data link layer protocols, HDLC, Point to Point Protocol.
Media Access control: Random Access, Controlled Access, Channelization.

Unit -3:

The Network Layer: Network layer design issues, Routing algorithms, Congestion control algorithms, Quality of service, Internetworking.
The network layer in the Internet: IPV4 Addresses, IPV6, Internet Control protocol, OSPF, BGP, IP, ICMPv4, IGMP.

Unit -4:

The Transport Layer: The Transport Service, Elements of Transport Protocols, Congestion Control, The internet transport protocols: UDP, TCP, Performance problems in computer networks, Network performancemeasurement.
The Application Layer: Introduction, Client-Server Programming, WWW and HTTP, FTP, e-mail, TELNET, Secure Shell, Domain Name System, SNMP

Unit -5:

Security Concepts: Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks.
Cryptography Concepts and Techniques: Introduction, plain text and cipher text, substitution techniques, transposition techniques.
E-Mail Security: Pretty Good Privacy, S/MIME. **IP Security:** IP Security overview, IP Security architecture.

Text Books:

1. Data communications and networking‖, Behrouz A. Forouzan, Mc Graw Hill Education, 5th edition, 2017.
2. Computer Networks‖, Andrew S. Tanenbaum, Wetherall, Pearson, 5th edition, 2011.
3. William Stallings, “Cryptography and Network Security”, 7th Edition, Pearson Education, 2016.

References:

1. Data Communication and Networks, Bhushan Trivedi, Oxford, 2016
2. Internetworking with TCP/IP – Principles, protocols, and architecture - Volume 1, Douglas E. Comer,5th edition, PHI, 2015
3. Computer Networks‖, 5E, Peterson, Davie, Elsevier.
4. Introduction to Computer Networks and Cyber Security‖, Chawan- Hwa Wu, Irwin, CRC Publications.
5. Computer Networks and Internets with Internet Applications‖, Comer.

Mapping of course outcomes with program outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3												2	
CO2	2	3											2	
CO3	2	2	3		1								2	
CO4	2	3	3		2								3	2
CO5	2	2											1	

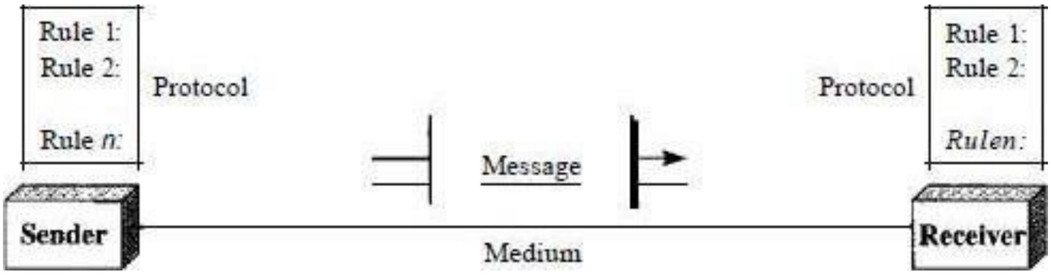
(Levels of Correlation, viz., 1-Low, 2-Moderate, 3 High)

UNIT-1
Introduction

DATA COMMUNICATIONS

The process of exchanging data or information between a source and receiver is called data communication.
Ex: When we communicate, we are sharing information. This sharing can be local or remote. Between individuals, local communication usually occurs face to face, while remote communication takes place over distance.

Components: A data communications system has five components.



1. Message: The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. Sender: The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
3. Receiver: The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. Transmission medium: The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
5. Protocol: A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

Data Representation:
Information today comes in different forms such as text, numbers, images, audio, and video.

Text*:* In data communications, text is represented as a bit pattern, a sequence of bits (Os or Is). Different sets of bit patterns have been designed to represent text symbols. Each set is called a code, and the process of representing symbols is called coding. Today, the prevalent coding system is called Unicode, which uses 32 bits to represent a symbol or character used in any language in the world. The American Standard Code for Information Interchange (ASCII), developed some decades ago in the United States, now constitutes the first 127 characters in Unicode and is also referred to as Basic Latin.

Numbers:

Numbers are also represented by bit patterns. However, a code such as ASCII is not used to represent numbers; the number is directly converted to a binary number to simplify mathematical operations.

Images:

Images are also represented by bit patterns. In its simplest form, an image is composed of a matrix of pixels (picture elements), where each pixel is a small dot. The size of the pixel depends on the *resolution.* For example, an image can be divided into 1000 pixels or 10,000 pixels.

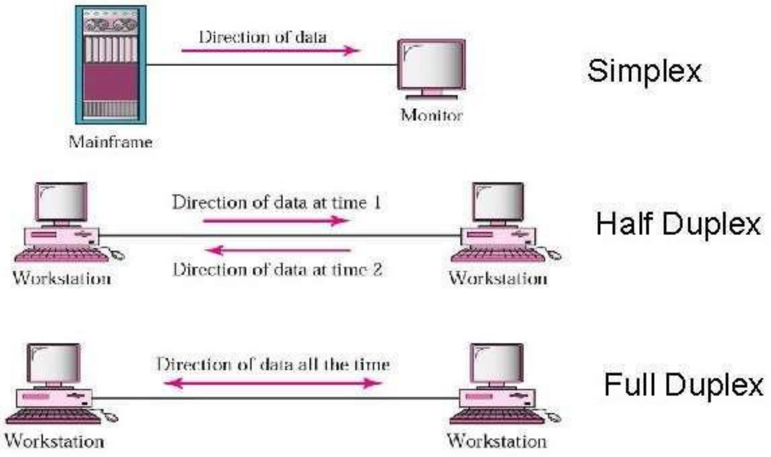
Audio:

Audio refers to the recording or broadcasting of sound or music. Audio is by nature different from text, numbers, or images. It is continuous, not discrete. Even when we use a microphone to change voice or music to an electric signal, we create a continuous signal.

Video:

Video refers to the recording or broadcasting of a picture or movie. Video can either be produced as a continuous entity (e.g., by a TV camera), or it can be a combination of images, each a discrete entity, arranged to convey the idea of motion.

Data Flow : Communication between two devices can be simplex, half-duplex, or full-duplex as shown in the Figure.



Simplex:

In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive. Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output. The simplex mode can use the entire capacity of the channel to send data in one direction.

Half-Duplex:

In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa . The half-duplex mode is like a one-lane road with traffic allowed in both directions.

In a half-duplex transmission, the entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time. Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

Full-Duplex:

In full-duplex both stations can transmit and receive simultaneously. The full-duplex mode is like a two way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link with signals going in the other direction. One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time. The full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.

NETWORKS

A **Computer network** to mean a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection need not be via a copper wire; fiber optics, microwaves, infrared, and communication satellites can also be used. The computers are autonomous, which are not forcibly started or controlled by other one. A system with one control unit and many slaves is not a computer network.

A computer network consists of end systems (or) nodes which are capable of transmitting of information, and which communicate through transit system interconnecting them. The transit system is also called an ***interconnection subsystem*** or simply a ***subnetwork***. An end system comprises of terminals, software and peripherally forming an autonomous system capable of performing information processing.

- **The old model of a single computer serving all of the organization's computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These systems are called computer networks.**

Network Criteria:

A network must be able to meet a certain number of criteria. The most important of these are *performance*, *reliability*, and *security*.

Performance: It can be measured in many ways, including *transit time* and *response time*.

- Transit time is the amount of time required for a message to travel from one device to another.
- Response time is the elapsed time between an inquiry and a response.

The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

Performance is often evaluated by two networking metrics: **throughput** and **delay**.

Reliability:

Network reliability is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network’s robustness in a catastrophe.

Security:

Network security issues include protecting data from unauthorized access, protecting data from damage. Developing and implementing policies and procedures for recovery from breaches and data losses.

Physical Structures:

Types of Connection

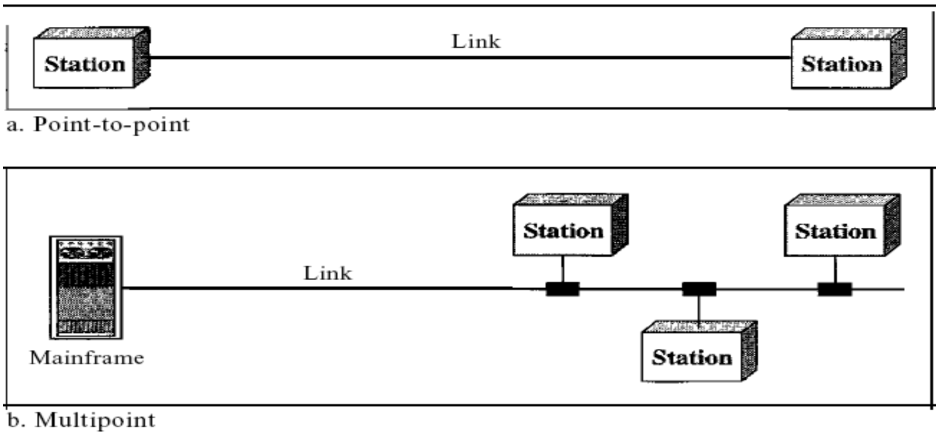
A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another. There are two possible types of connections: point-to-point and multipoint.

Point-to-Point:

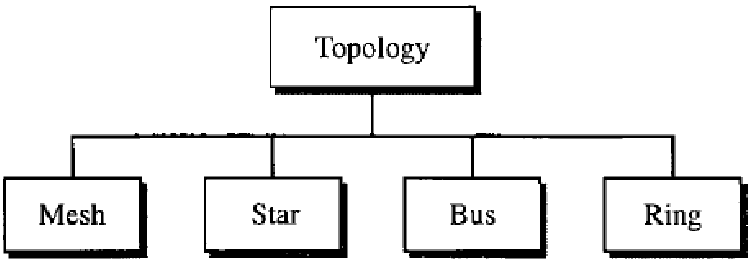
A **point-to-point connection** provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices.

Multipoint:

A **multipoint** (also called **multidrop**) **connection** is one in which more than two specific devices share a single link. In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a ***spatially shared*** connection. If users must take turns, it is a ***timeshared*** connection.



Physical Topology



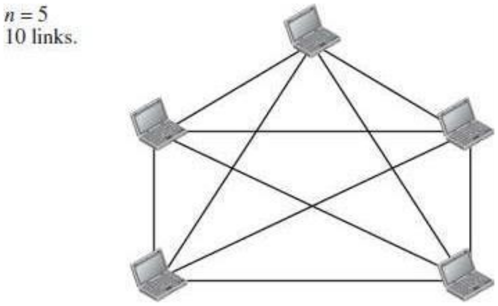
The term ***physical topology*** refers to the way in which a network is laid out physically. A **network topology** is the arrangement of a **network**, including its nodes and connecting lines. There are two ways of **defining network** geometry: the **physical topology** and the **logical** (or signal) **topology**. There are four basic topologies possible: mesh, star, bus, and ring.

Mesh Topology

In a **mesh topology,** every device has a dedicated point-to-point link to every other device. The term *dedicated* means that the link carries traffic only between the two devices it connects.

- In a mesh topology, we need ***n* (*n* – 1) / 2** duplex-mode links.

Every device on the network must have *n* – 1 input/output (I/O) ports to be connected to the other *n* – 1 station.



Advantages:

- The use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices.
- A mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system.
- There is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages.
- Finally, point-to-point links make fault identification and fault isolation easy.

Disadvantages:

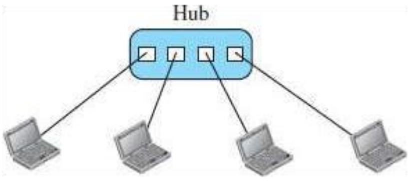
- Installation and reconnection are difficult.
- The sheer bulk of the wiring can be greater than the available space can accommodate.
- The hardware required to connect each link (I/O ports and cable) can be prohibitively expensive.

Ex: connection of telephone regional offices.

Star Topology

In a **star topology,** each device has a dedicated point-to-point link only to a central controller, usually called a ***hub.*** The devices are not directly linked to one another. A star topology does not allow direct traffic between devices.

The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device.



Advantages:

- Less expensive than a mesh topology.
- Easy to install and reconfigure.
- Additions, moves, and deletions involve only one connection: between that device and the hub.

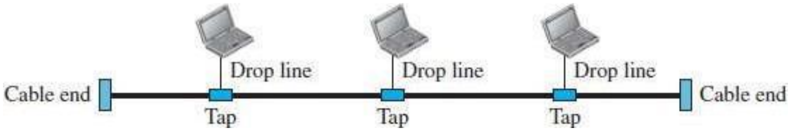
- Star topology is robust.
- If one link fails, only that link is affected. All other links remain active. This factor also lends itself to easy fault identification and fault isolation.
- Hub can be used to monitor link problems and bypass defective links.

Disadvantage:

- Star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead.

Bus Topology

A **bus topology** is example of multipoint Link. One long cable acts as a **backbone** to link all the devices in a network.



Nodes are connected to the bus cable by *drop lines* and *taps*. A *drop line* is a connection running between the device and the main cable. A *tap* is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core.

Advantages:

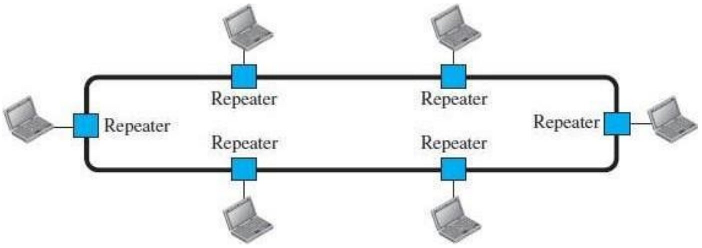
- Ease of installation.
- A bus uses less cabling than mesh or star topologies.
- Only the backbone cable stretches through the entire facility

Disadvantages:

- It includes difficult reconnection and fault isolation.
- Difficult to add new devices.
- Signal reflection at the taps can cause degradation in quality.
- Fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

Ring Topology

In a **ring topology,** each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater.



Advantages:

- A ring is relatively easy to install and reconfigure.
- To add or delete a device requires changing only two connections.
- Fault isolation is simplified.

Disadvantages:

- Unidirectional traffic can be a disadvantage.
- A break in the ring (such as a disabled station) can disable the entire network.

NETWORK TYPES

One network can be distinguished from another network based on few criteria such as size, Geographical area, and ownership. There are 3 basic types of Networks. They are Local area networks and Wide Area Networks

. ***Local Area Network (LAN):*** **LAN'**s, are privately-owned networks within a single building or campus of up to a few kilometers in size. They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information.

- Each host in a LAN has an identifier, an address that uniquely defines the host in the LAN.
- A packet sent by a host to another host carries both the source host's and the destination host's addresses.

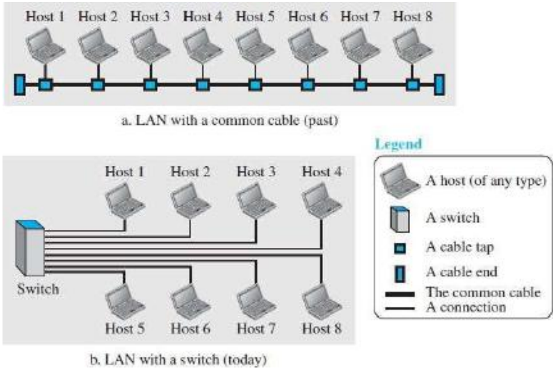
LANs are distinguished from other kinds of networks b y three characteristics:

1) Their size
2) Their transmission technology, and
3) Their topology.

I. ***Size:*** LANs are restricted in size, which means that the worst -case transmission time is bounded and known in advance.

II. **Transmission technology:** LANs consisting of a cable to which all the machines are attached. Traditional LANs run at speeds of 10 Mbps to 100 Mbps, have low delay (microseconds or nanoseconds), and make very few errors. NewerLANs operate at up to 10 Gbps. [(1 Mbps is 1,000,000 bits/sec) and gigabits/sec (1 Gbps is 1,000,000,000 bits/sec)].

Topology: Various topologies are possible for broadcast

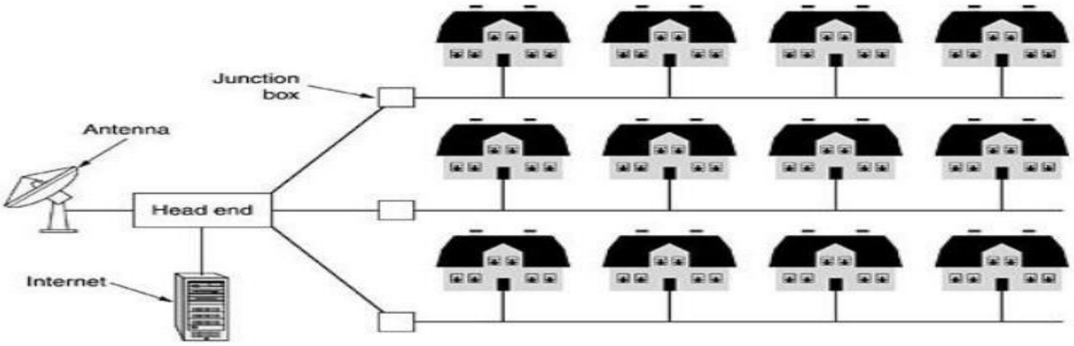


a. LAN with a common cable (past)

b. LAN with a switch (today)

Ex: Bus and Ring.

Metropolitan Area Network (MAN): MAN, covers a city. The best-known example of a MAN is the cable television network available in many cities. This system grew from earlier community antenna systems used in areas with poor over-the-air television reception. In these early systems, a large antenna was placed on top of a nearby hill and signal was then piped to the subscribers' houses. At first, these were locally-designed, ad hoc systems. Then companies began jumping into the business, getting contracts from city governments to wire up an entire city.



Wide Area Network:

- A **wide area network (WAN)** is also an interconnection of devices capable of communication.
- A wide area network, or WAN, spans a large geographical area, often a country or continent.
- It contains a collection of machines intended for running user (i.e., application) programs.
- A WAN interconnects connecting devices such as switches, routers, or modems.
- A WAN is normally created and run by communication companies and leased by an organization that uses it.

Ex: point-to-point WANs and switched WANs.

The collections of machines called as *hosts.* The hosts are connected by a *communicationsubnet*, or just subnet for short.

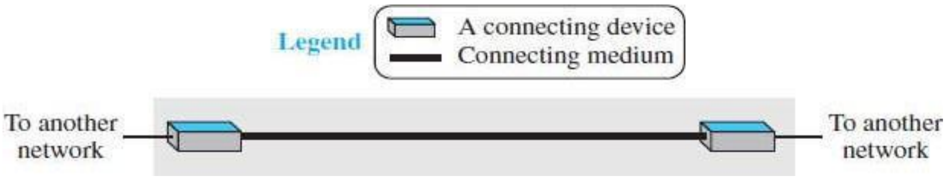
- The job of the subnet is to carry messages from host to host.
- In most wide area networks, the subnet consists of two distinct components: ***transmission lines*** *and switching elements.*

  - **Transmission lines:** move bits between machines. They can be made of copper wire, optical fiber, or even radio links.
  - **Switching elements:** These are specialized computers that connect three or more transmission lines. When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them. These switching computers have been called as *Router.*

❖ The collection of communication lines and routers (but not the hosts) form the subnet.

i. Point-to-Point WAN:

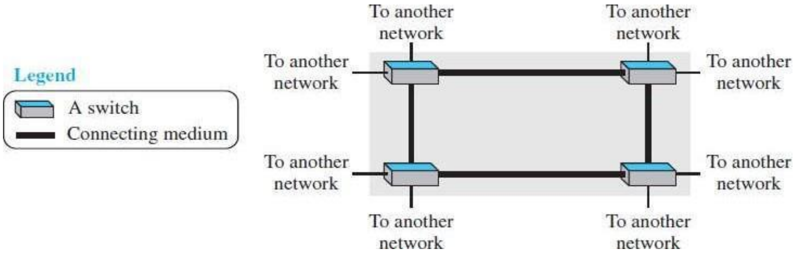
A point-to-point WAN is a network that connects two communicating devices through a



transmission media (cable or air).

ii. Switched WAN:

A switched WAN is a network with more than two ends. Switched WAN is a combination of several point- to-point WANs that are connected by switches.



Internetwork:

When two or more networks are connected, they make an **internetwork,** or **internet.**

A collection of interconnected networks is called an ***internetwork*** or ***internet***.

- Internet is a collection of LANs connected by a WAN.

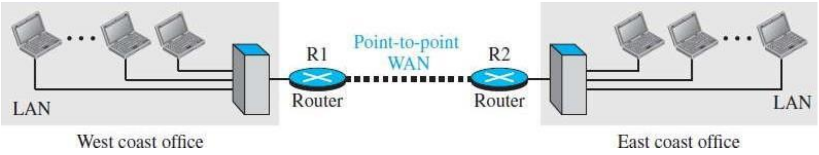


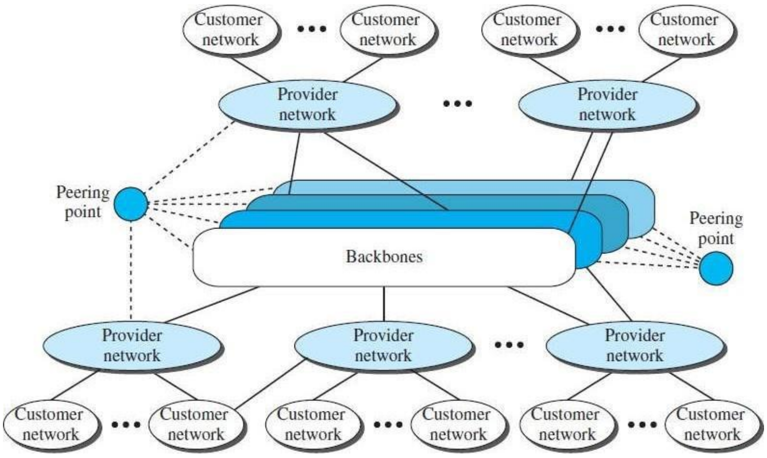
Fig: *An internetwork made of two LANs and one point-to-point WAN*

When a host in the west coast office sends a message to another host in the same office, the router blocks the message, but the switch directs the message to the destination. On the other hand, when a host on the west coast sends a message to a host on the east coast, router R1 routes the packet to router R2, and the packet reaches the destination.

<u>The Internet</u>

Internet is composed of thousands of interconnected networks. The figure shows the Internet as several backbones, provider networks, and customer networks.

- The *backbones* are large networks are connected through some complex switching systems, called *peering points.*
- *Provider networks* are smaller networks that use the services of the backbones for a fee.
  The *customer networks* use the services provided by the Internet
- Backbones and provider networks are also called **Internet Service Providers (ISPs).** The backbones are often referred to as *international ISPs;* the provider networks are often referred to as *national* or *regional ISPs*.



Accessing the Internet

Using Telephone Networks:

One option for residences and small businesses to connect to the Internet is to change the voice line between the residence or business and the telephone center to a point-to-point WAN. This can be done in two ways.

- ***Dial-up service:*** The first solution is to add to the telephone line a modem that converts data to voice.
- ***DSL Service:*** The DSL service also allows the line to be used simultaneously for voice and data communication.

Using Cable Networks:

A residence or a small business can be connected to the Internet by using this service. It provides a higher speed connection, but the speed varies depending on the number of neighbors that use the same cable.

Using Wireless Networks:

Wireless connectivity has recently become increasingly popular. With the growing wireless WAN access, a household or a small business can be connected to the Internet through a wireless WAN.

Direct Connection to the Internet:

A large organization or a large corporation can itself become a local ISP and be connected to the Internet. This can be done if the organization or the corporation leases a high-speed WAN from a carrier provider and connects itself to a regional ISP. For example, a large university with several campuses can create an internetwork and then connect the internetwork to the Internet.

INTERNET HISTORY

Early History

Before 1960, there were telegraph and telephone networks, suitable for constant-rate communication at that time, which means that after a connection was made between two users, the encoded message (telegraphy) or voice (telephony) could be exchanged. To handle bursty data we needed to invent packet-switched network. ***Birth of Packet-Switched Networks***

The theory of packet switching for bursty traffic was first presented by Leonard Kleinrock in 1961 at MIT.

ARPANET

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for the **Advanced Research Projects Agency Network (ARPANET),** a small network of connected computers. The idea was that:

- Each host computer would be attached to a specialized computer, called an *interface message processor* (IMP).
- The IMPs, in turn, would be connected to each other.
- Each IMP had to be able to communicate with other IMPs as well as with its own
- attached host. By 1969, ARPANET was a reality.
- *Network Control Protocol* (NCP) provided communication between the hosts.

Birth of the Internet

To link dissimilar networks, there were many problems to overcome: diverse packet sizes, diverse interfaces, and diverse transmission rates, as well as differing reliability requirements. *Cerf* and *Kahn* devised the idea of a device called a *gateway* to serve as the intermediary hardware to transfer data from one network to another.

TCP/IP

In 1973 Cerf and Kahn outlined the protocols transmission control protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway.

A radical idea was the transfer of responsibility for *error correction* from the IMP to the host machine. In TCP/IP, IP would handle datagram routing while TCP would be responsible for higher level functions such as segmentation, reassembly, and error detection

In 1981, under a Defence Department contract, UC Berkeley modified the UNIX operating system to include TCP/IP but it did much for the popularity of internetworking.

In 1983, TCP/IP became the official protocol for the ARPANET.

MILNET

In 1983, ARPANET split into two networks: **Military Network (MILNET)** for military users and ARPANET for nonmilitary users.

CSNET

Computer Science Network (CSNET) was created in 1981 and it was sponsored by National Science Foundation (NSF).

NSFNET

With the success of CSNET, the NSF in 1986 sponsored the **National Science Foundation Network (NSFNET),** a backbone that connected five supercomputer centers located throughout the United States.

ANSNET

In 1991, the U.S. government decided that NSFNET was not capable of supporting the rapidly increasing Internet traffic. Three companies, IBM, Merit, and Verizon, filled the void by forming a nonprofit organization called Advanced Network & Services (ANS) to build a new, high-speed

Internet backbone called **Advanced Network Services Network (ANSNET).**

Internet Today

The Internet today is a set of pier networks that provide services to the whole world.

World Wide Web

The Web was invented at CERN by Tim Berners-Lee. This invention has added the commercial applications to the Internet.

Multimedia

Recent developments in the multimedia applications such as voice over IP (telephony), video over IP (Skype), view sharing (YouTube), and television over IP (PPLive) has increased the number of users and the amount of time each user spends on the network.

Peer-to-Peer Applications

Peer-to-peer networking is also a new area of communication with a lot of potential.

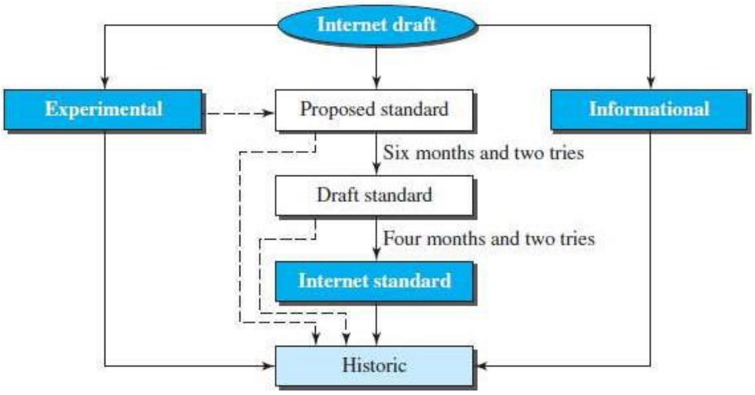
STANDARDS AND ADMINISTRATION

An **Internet standard** is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status. A specification begins as an Internet draft.

An **Internet draft** is a working document with no official status and a six -month lifetime.

A draft may be published as a **Request for Comment (RFC).** Each RFC is edited, assigned a number, and made available to all interested parties.

Figure 1.16 *Maturity levels of an RFC*



Maturity Levels

An RFC, during its lifetime, falls into one of six *maturity levels:* proposed standard, draft standard, Internet standard, historic, experimental, and informational.

- ***Proposed Standard***: A proposed standard is a specification that is stable, well understood, and of sufficient interest to the Internet community. At this level, the specification is usually tested and implemented by several different groups.
- ***Draft Standard:*** A proposed standard is elevated to draft standard status after at least two successful independent and interoperable implementations. Barring difficulties, a draft standard, with modifications if specific problems are encountered, normally becomes an Internet standard.

- ***Internet Standard:*** A draft standard reaches Internet standard status after demonstrations of successful implementation.
- ***Historic:*** The historic RFCs are significant from a historical perspective. They either have been superseded by later specifications or have never passed the necessary maturity levels to become an Internet standard.
- ***Experimental:*** An RFC classified as experimental describes work related to an experimental situation that does not affect the operation of the Internet. Such an RFC should not be implemented in any functional Internet service.
- ***Informational:*** An RFC classified as informational contains general, historical, or tutorial information related to the Internet. It is usually written by someone in a non-Internet organization, such as a vendor.

Requirement Levels

RFCs are classified into five *requirement levels:* required, recommended, elective, limited use, and not recommended.

- ***Required:*** An RFC is labeled *required* if it must be implemented by all Internet systems to achieve minimum conformance. For example, IP and ICMP (Chapter 19) are req uired protocols
- ***Recommended:*** An RFC labeled recommended is not required for minimum conformance; it is recommended because of its usefulness. For example, FTP and TELNET.
- ***Elective:*** An RFC labeled elective is not required and not recommended. However, a system can use it for its own benefit.
- ***Limited Use:*** An RFC labeled limited use should be used only in limited situations.
- ***Not Recommended.*** An RFC labeled not recommended is inappropriate for general use. Normally a historic (deprecated) RFC may fall under this category

Internet Administration

General organization of Internet administration is:

ISOC

The **Internet Society (ISOC)** is an international, nonprofit organization formed in 1992 to provide support for the Internet standards process. ISOC accomplishes this through maintaining and supporting other Internet administrative bodies such as IAB, IETF, IRTF, and IANA. ISOC also promotes research and other scholarly activities relating to the Internet.

IAB

The **Internet Architecture Board (IAB)** is the technical advisor to the ISOC. The main purposes of the IAB are to oversee the continuing development of the TCP/IP Protocol Suite and to serve in a technical advisory capacity to research members of the Internet community. IAB accomplishes this through its two primary components, the Internet Engineering Task Force (IETF) and the Internet Research Task Force (IRTF).

IETF

The **Internet Engineering Task Force (IETF)** is a forum of working groups managed by the Internet Engineering Steering Group (IESG). IETF is responsible for identifying operational problems and proposing solutions to these problems. IETF also develops and reviews specifications intended as Internet standards.

IRTF

The **Internet Research Task Force (IRTF)** is a forum of working groups managed by the Internet Research Steering Group (IRSG). IRTF focuses on long-term research topics related to Internet protocols, applications, architecture, and technology.

NETWORK MODELS

Protocol Hierarchies

To reduce their design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. The name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it. This concept is actually a familiar one and is used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

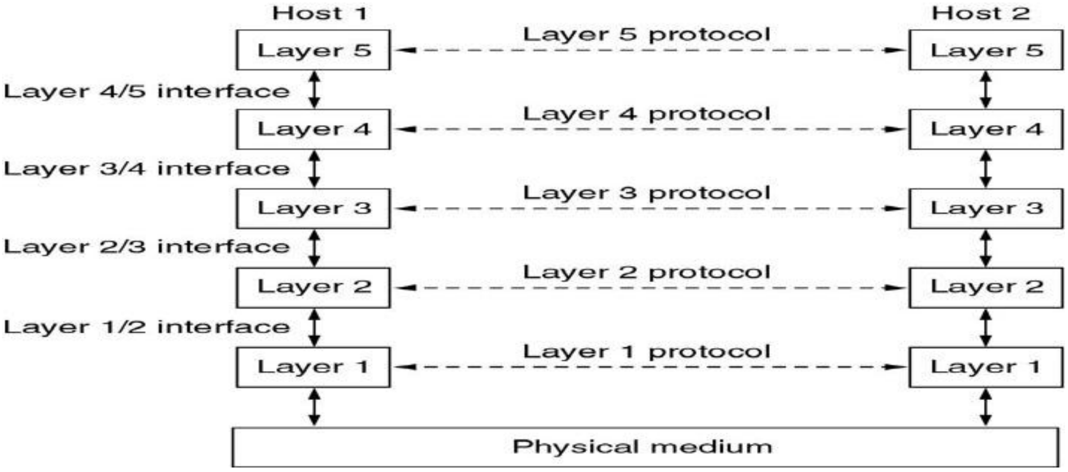
When layer 'n' on one machine carries on a conversation with layer 'n' on another machine, the rules and conventions used in this conversation are collectively known as the layer n protocol. Basically, a protocol is an agreement between the communicating parties on how communication is to proceed. Violating the protocol will make communication more difficult, if not completely impossible.

A five-layer network is illustrated.

The entities comprising the corresponding layers on different machines are called peers.

The peers may be software processes, hardware devices, or even human beings.

In other words, it is the peers that communicate by using the protocol to talk to each other.



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.

Virtual communication is shown by dotted lines and physical communication by solid lines. Between each pair of adjacent layers is an interface. The interface defines which primitive operations and services the lower layer makes available to the upper one.

When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is defining clean interfaces between the layers. Doing so, in turn, requires that each layer perform a specific collection of well-understood functions.

In addition to minimizing the amount of information that must be passed between layers, clear cut interfaces also make it simpler to replace one layer with a completely different protocol or implementation (e.g., replacing all the telephone lines by satellite channels) because all that is required of the new protocol or implementation is that it offer exactly the same set of services to its upstairs neighbor as the old one did. It is common that different hosts use different implementations of the same protocol (often written by different companies). In fact, the protocol itself can change in some layer without the layers above and below it even noticing.

A set of layers and protocols is called <u>network architecture</u>.

The specification of architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification of the interfaces is part of the architecture because these are hidden away inside the machines and not visible from the outside.It is not even necessary that the interfaces on all machines in a network be the same, provided that each machine can correctly use all the protocols.

A list of the protocols used by a certain system, one protocol per layer, is called a <u>protocol stack</u>.

<u>Design Issues for the Layers</u>

<u>1.Reliability</u>: It is the design issue of making a network that operates correctly even though it is made up of a collection of components that are themselves unreliable. Think about the bits of a packet traveling through the network. There is a chance that some of these bits will be received damaged (inverted) due to fluke electrical noise, random wireless signals, hardware flaws, software bugs and so on. How is it possible that we find and fix these errors?

One mechanism for finding errors in received information uses <u>codes for error detection</u>. Information that is incorrectly received can then be retransmitted until it is received correctly. More powerful codes allow for <u>error correction</u>, where the correct message is recovered from the possibly incorrect bits that were originally received. Both of these mechanisms work by adding redundant information. They are used at low layers, to protect packets sent over individual links, and high layers, to check that the right contents were received.

Another reliability issue is finding a working path through a network. Often there are multiple paths between a source and destination, and in a large network, there may be some links or routers that are broken. The network should automatically make this decision. This is called routing.

2.The evolution of the network: Over time, networks grow larger and new designs emerge that need to be connected to the existing network. We have recently seen the key structuring mechanism used to support change by dividing the overall problem and hiding implementation details: protocol layering. There are many other strategies as well.

Since there are many computers on the network, every layer needs a mechanism for identifying the senders and receivers that are involved in a particular message. This mechanism is called addressing or naming, in the low and high layers, respectively.

An aspect of growth is that different network technologies often have different limitations. For example, not all communication channels preserve the order of messages sent on them, leading to solutions that number messages. Another example is differences in the maximum size of a message that the networks can transmit. This leads to mechanisms for disassembling, transmitting, and then reassembling messages. This overall topic is called internetworking.

When networks get large, new problems arise. Cities can have traffic jams, a shortage of telephone numbers, and it is easy to get lost. Not many people have these problems in their own neighborhood, but citywide they may be a big issue.

Designs that continue to work well when the network gets large are said to be scalable.

3. Resource allocation.

Networks provide a service to hosts from their underlying resources, such as the capacity of transmission lines. To do this well, they need mechanisms that divide their resources so that one host does not interfere with another too much. Many designs share network bandwidth dynamically, according to the short term needs of hosts, rather than by giving each host a fixed fraction of the bandwidth that it may or may not use. This design is called statistical multiplexing, meaning sharing based on the statistics of demand. It can be applied at low layers for a single link, or at high layers for a network or even applications that use the network. An allocation problem that occurs at every level is how to keep a fast sender from swamping a slow receiver with data. Feedback from the receiver to the sender is often used. This is called flow control.

Sometimes the problem is that the network is oversubscribed because too many computers want to send too much traffic, and the network cannot deliver it all. This overloading of the network is called congestion.

One strategy is for each computer to reduce its demand when it experiences congestion. It, too can be used in all layers. It is interesting to observe that the network has more resources to offer than simply bandwidth. For uses such as carrying live video, the timeliness of delivery matters a great deal. Most networks must provide service to applications that want this real-time delivery at the same time that they provide service to applications that want high throughput.

Quality of service is the name given to mechanisms that reconcile these competing demands.

4. Security: The last major design issue is to secure the network by defending it against different kinds of threats. One of the threats we have mentioned previously is that of eavesdropping on communications. Mechanisms that provide confidentiality defend against this threat, and they are used in multiple layers. Mechanisms for authentication prevent someone from impersonating someone else. They might be used to tell fake banking Web sites from the real one, or to let the cellular network check that a call is really coming from your phone so that you will pay the bill. Other mechanisms for integrity prevent surreptitious changes to messages, such as altering ‘‘debit my account $10’’ to ‘‘debit my account $1000.’’

Connection-Oriented Versus Connectionless Service

Layers can offer two different types of service to the layers above them: connection-oriented and connectionless.

Connection-oriented 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 at the other end. In most cases the order is preserved so that the bits arrive in the order they were sent. In some cases when a connection is established, the sender, receiver, and subnet conduct a negotiation about the parameters to be used, such as maximum message size, quality of service required, and other issues. Typically, one side makes a proposal and the other side can accept it, reject it, or make a counterproposal.

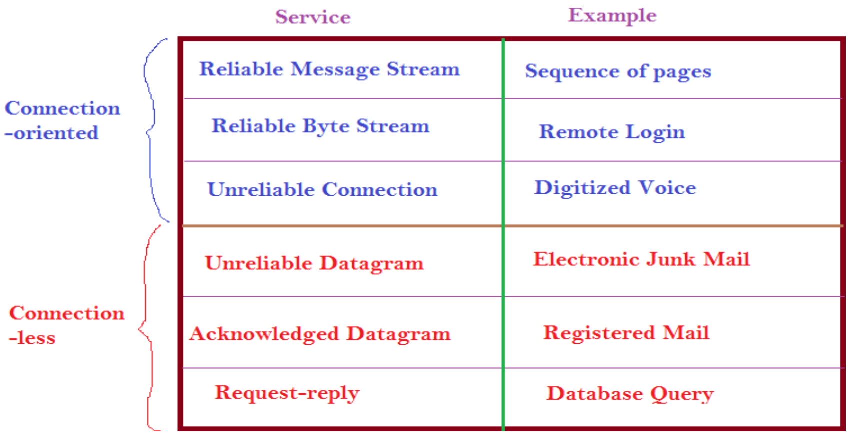
In contrast to connection-oriented service, connectionless service is modeled after the postal system. Each message (letter) carries the full destination address, and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages. When the intermediate nodes receive a message in full before sending it on to the next node, this is called store-and-forward switching.

The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called cut-through switching.

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 kind of service can further be characterized by its reliability. Some services are reliable in the sense that they never lose data.

Not all applications require connections. For example, spammers send electronic junk-mail to many recipients. The spammer probably does not want to go to the trouble of setting up and later tearing down a connection to a recipient just to send them one item. Unreliable (meaning not acknowledged) connectionless service is often called datagram service, in analogy with telegram service, which also does not return an acknowledgement to the sender. Despite it being unreliable, it is the dominant form in most networks for reasons that will become clear later. 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. For example, a mobile phone client might send a query to a map server to retrieve the map data for the current location.



	Service	Example
Connection -oriented	Reliable Message Stream	Sequence of pages
	Reliable Byte Stream	Remote Login
	Unreliable Connection	Digitized Voice
Connection -less	Unreliable Datagram	Electronic Junk Mail
	Acknowledged Datagram	Registered Mail
	Request-reply	Database Query

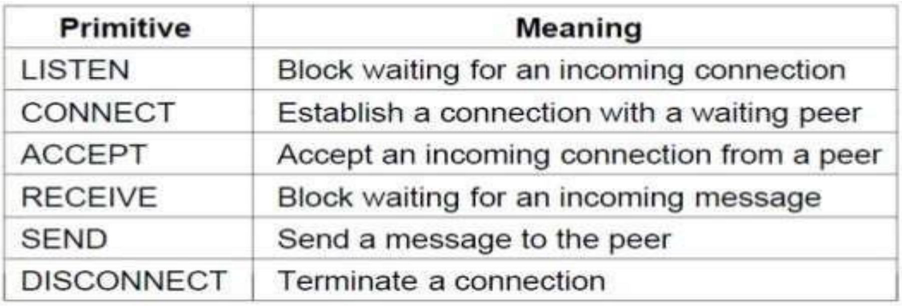
The concept of using unreliable communication may be confusing at first.

After all, why would anyone actually prefer unreliable communication to reliable communication? First of all, reliable communication (in our sense that is, acknowledged) may not be available in a given layer. For example, Ethernet does not provide reliable communication. Packets can occasionally be damaged in transit. It is up to higher protocol levels to recover from this problem. In particular, many reliable services are built on top of an unreliable datagram service. Second, the delays inherent in providing a reliable service may be unacceptable, especially in real-time applications such as multimedia. For these reasons, both reliable and unreliable communication coexists.

Service Primitives

A service is formally specified by a set of primitives (operations) available to user processes to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets.

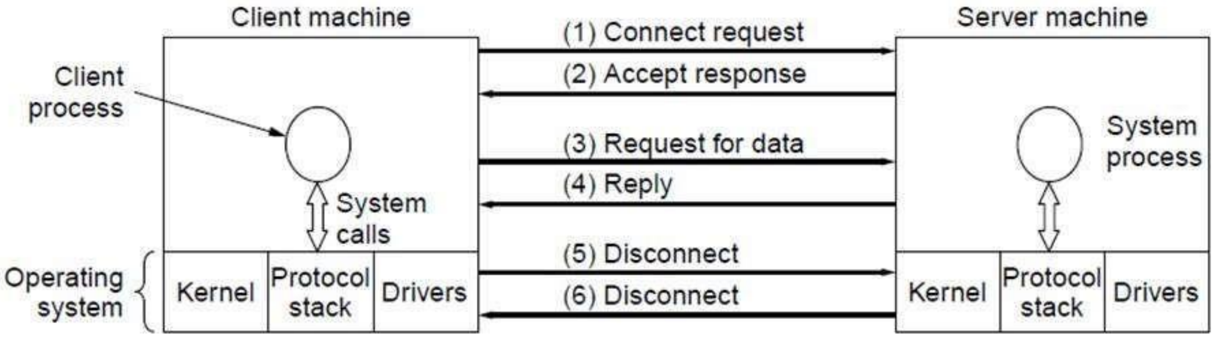
The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connectionless service.



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

First, the server executes LISTEN to indicate that it is prepared to accept incoming connections. A common way to implement LISTEN is to make it a blocking system call. After executing the primitive, the server process is blocked until a request for connection appears.

Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server’s address. The operating system then typically sends a packet to the peer asking it to connect (1). The client process is suspended until there is a response. When the packet arrives at the server, the operating system sees that the packet is requesting a connection. It checks to see if there is a listener, and if so it unblocks the listener.



The server process can then establish the connection with the ACCEPT call. This sends a response (2) back to the client process to accept the connection. The arrival of this response then releases the client. At this point the client and server are both running and they have a connection established. The obvious analogy between this protocol and real life is a customer (client) calling a company's customer service manager. At the start of the day, the service manager sits next to his telephone in case it rings. Later, a client places a call. When the manager picks up the phone, the connection is established.

The next step is for the server to execute RECEIVE to prepare to accept the first request. Normally, 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.

Then the client executes SEND to transmit its request (3) followed by the execution of RECEIVE to get the reply. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the serve r uses SEND to return the answer to the client (4).

The arrival of this packet unblocks the client, which can now inspect the answer. If the client has additional requests, it can make them now. When the client is done, it executes DISCONNECT to terminate the connection(5). Usually, an initial DISCONNECT is a blocking call, suspending the client and sending a packet to the server saying that the connection is no longer needed.

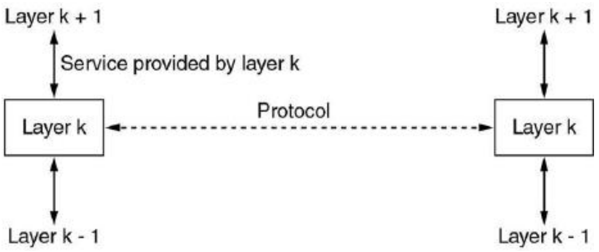
When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection (6). When the server's packet gets back to the client machine, the client process is released and the connection is broken. In a nutshell, this is how connection-oriented communication works..

The Relationship of Services to Protocols

A service is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A 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 packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users. In this way, the service and the protocol are completely decoupled. This is a key concept that any network designer should understand well. To repeat this crucial point, services relate to the interfaces between layers.

In contrast, protocols relate to the packets sent between peer entities on different machines. It is very important not to confuse the two concepts

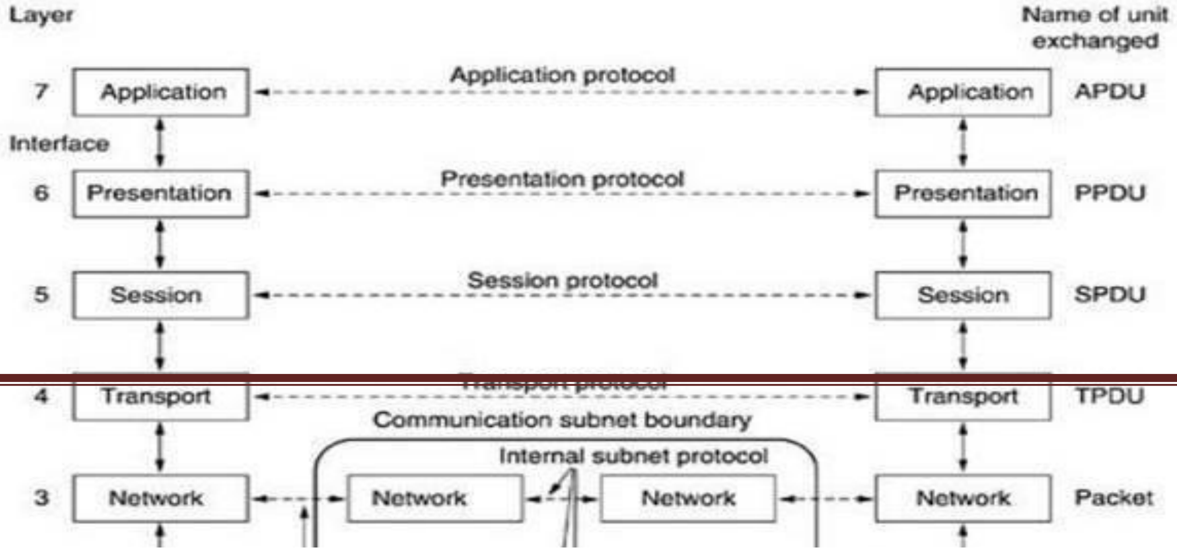


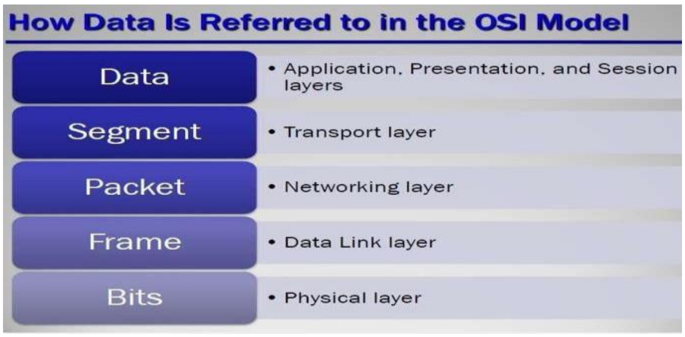
Reference Models

The OSI Model

This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983). It was revised in 1995(Day, 1995). The model is called the ISO-OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems.

The OSI model has seven layers. Each layer has specific functions. All layers work together in the correct order to move data around anetwork





How Data Is Referred to in the OSI Model

Data	• Application, Presentation, and Session layers
Segment	• Transport layer
Packet	• Networking layer
Frame	• Data Link layer
Bits	• Physical layer

Physical Layer

- Deals with all aspects of physically moving data from one computer to the next
- Converts data from the upper layers into 1s and 0s for transmission over media
- Defines how data is encoded onto the media to transmit the data
- Defined on this layer: Cable standards, wireless standards, and fiber optic standards.. Copper wiring, fiber optic cable, radio frequencies, anything that can be used to transmit data is defined on the Physical layer of the OSI Model
- Device example: Hub
- Used to transmit data

Data Link Layer

- Is responsible for moving frames from node to node or computer to computer
- Can move frames from one adjacent computer to another, cannot move frames across routers
- Encapsulation = frame
- Requires MAC address or *physical address*

- Protocols defined include Ethernet Protocol and Point-to-Point Protocol (PPP)
- Device example: Switch
- Two sublayers: Logical Link Control **(LLC)** and the Media Access Control **(MAC)**

  Logical Link Control (LLC)–Data Link layer addressing, flow control, address notification, errorcontrol

  Media Access Control (MAC)–Determines which computer has access to the network media at any given time, determines where one frame ends and the next one starts, called frame Synchronization

Network Layer

- Responsible for moving packets (data) from one end of the network to the other, called *end-to-end communications*
- Requires *logical addresses* such as IP addresses
- Device example: Router
- Routing is the ability of various network devices and their related software to move data packets from source to destination

Transport Layer

- Takes data from higher levels of OSI Model and breaks it into segments that can be sent to lower-level layers for data transmission
- Conversely, reassembles data segments into data that higher-level protocols

  and applications can use
- Also puts segments in correct order (called sequencing) so they can be reassembled in correct order at destination
- Concerned with the reliability of the transport of sent data
- May use a *connection-oriented protocol* such as TCP to ensure destination received segments
- May use a *connectionless protocol* such as UDP to send segments without assurance of delivery
- Uses port addressing

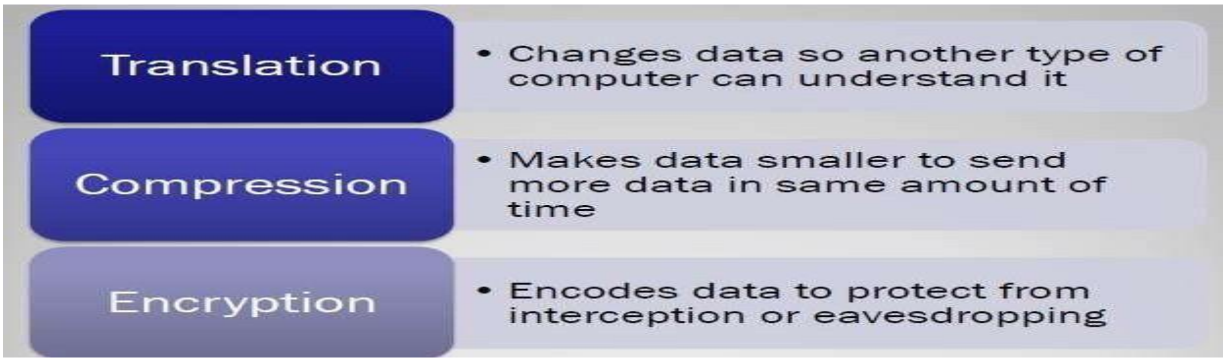
Session Layer

- Responsible for managing the dialog between networked devices

- Establishes, manages, and terminates connections
- Provides duplex, half-duplex, or simplex communications between devices
- Provides procedures for establishing checkpoints, adjournment, termination, and restart or recovery procedures

Presentation Layer

- Concerned with how data is presented to the network
- Handles three primary tasks: –Translation , –Compression , –Encryption



Translation
- Changes data so another type of computer can understand it

Compression
- Makes data smaller to send more data in same amount of time

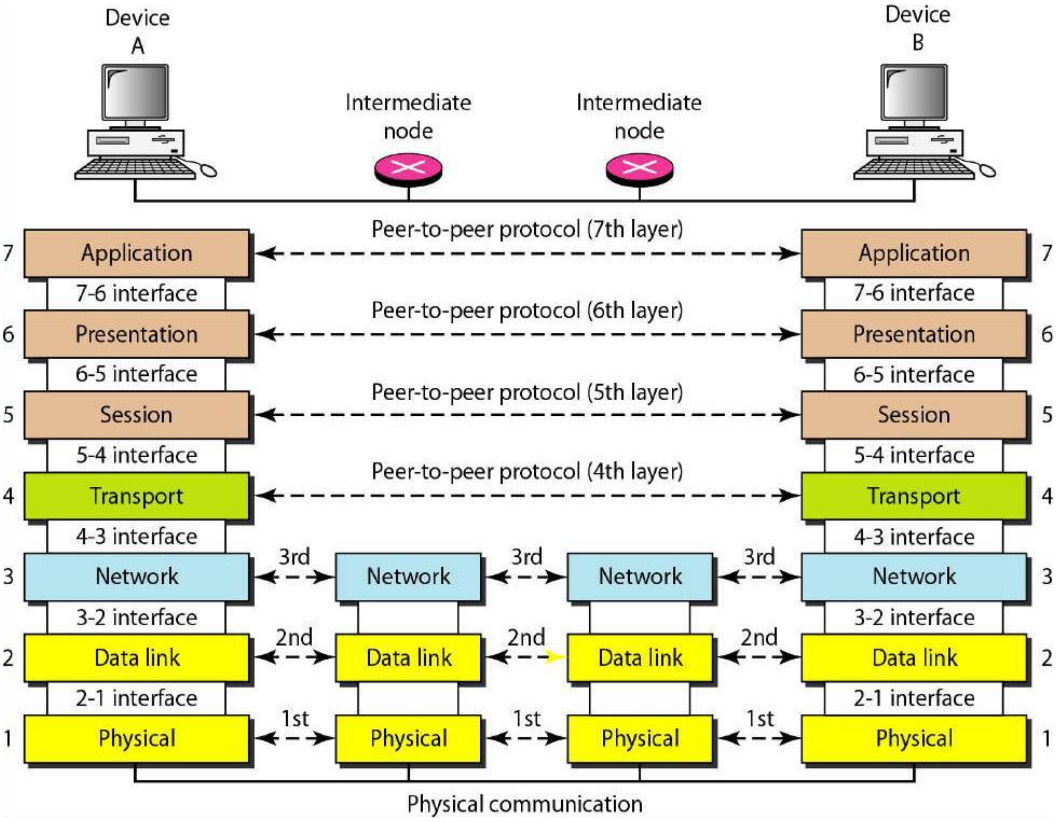
Encryption
- Encodes data to protect from interception or eavesdropping

Application Layer

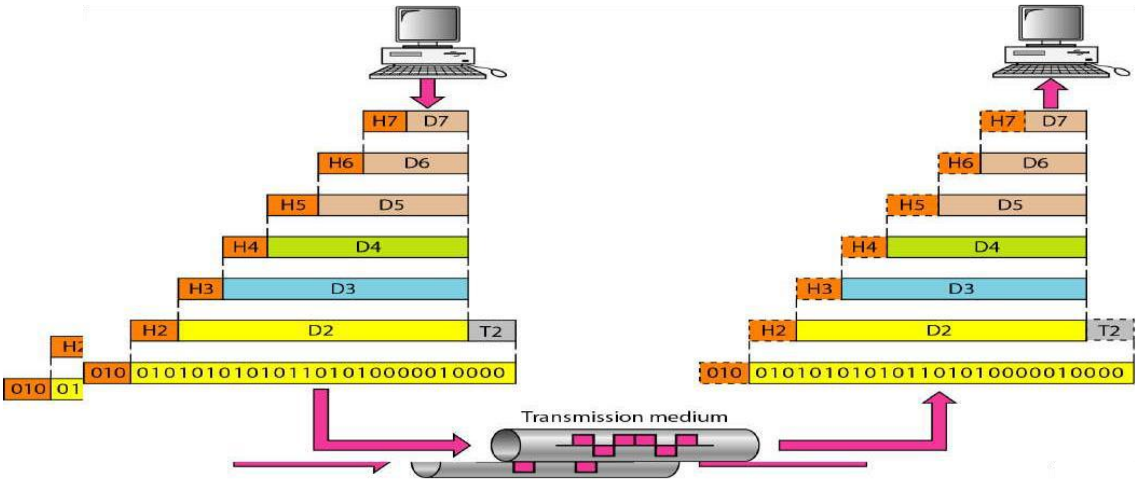
- Contains all services or protocols needed by application software or operating system to communicate on the network
- Examples

    Firefox web browser uses HTTP (Hyper-Text Transport Protocol)

    E-mail program may use POP3 (Post Office Protocol version 3) to read e-mails and SMTP (Simple Mail Transport Protocol) to send e-mails

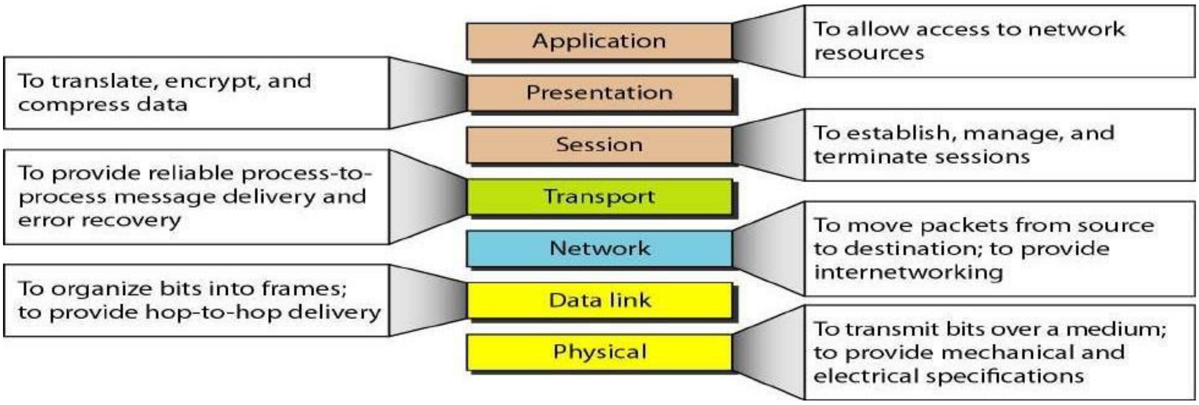


The interaction between layers in the OSI model



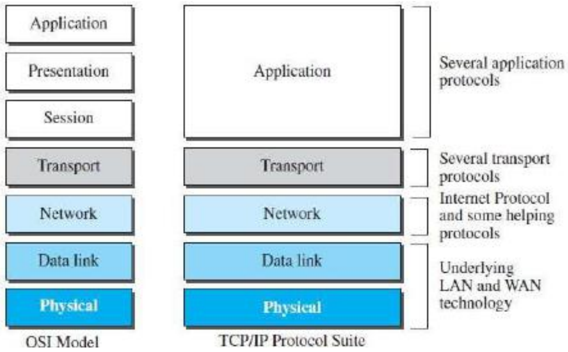
An exchange using the OSI model

Summary:


Application
To allow access to network resources
Presentation
To translate, encrypt, and compress data
Session
To establish, manage, and terminate sessions
Transport
To provide reliable process-to-process message delivery and error recovery
Network
To move packets from source to destination; to provide internetworking
Data link
To organize bits into frames; to provide hop-to-hop delivery
Physical
To transmit bits over a medium; to provide mechanical and electrical specifications

<u>OSI vs TCP/IP</u>:

Session and Presentation layers were not added to the TCP/IP protocol suite after the publication of the OSI model. The application layer in the suite is usually considered to be the combination of three layers in the OSI model.



Two reasons were mentioned for this decision.

First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols.

Second, the application layer is not only one piece of software. Many applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

Lack of OSI Model's Success

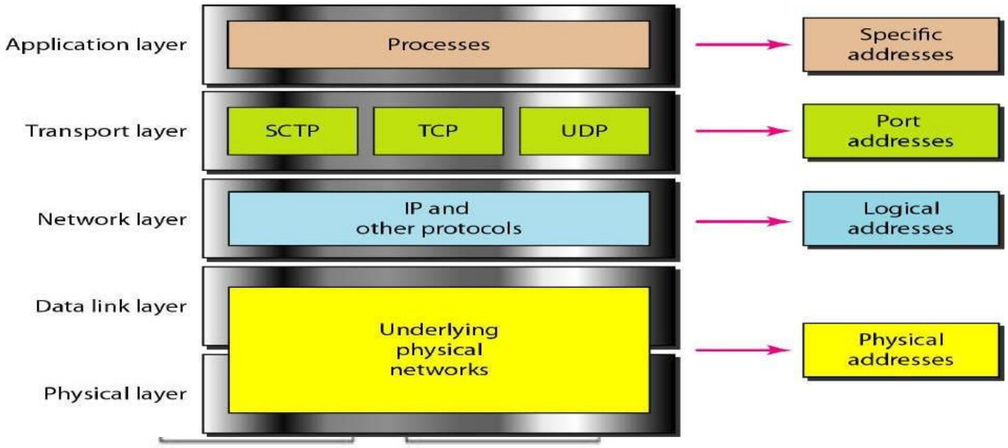
The OSI model appeared after the TCP/IP protocol suite. TCP/IP protocol cannot be fully replaced by the OSI for three reasons:

First, OSI was completed when TCP/IP was fully in place and a lot of time and money had been spent on the suite; changing it would cost a lot.

Second, some layers in the OSI model were never fully defined.

Third, when OSI was implemented by an organization in a different application, it did not show a high enough level of performance to entice the Internet authority to switch from the TCP/IP protocol suite to the OSI model.

<u>TCP/IP Model (Transmission Control Protocol/Internet Protocol)</u>–A *protocol suite* is a large number of related protocols that work together to allow networked computers to communicate



Relationship of layers and addresses in TCP/IP

<u>Application Layer</u>

- Application layer protocols define the rules when implementing specific network applications
- Rely on the underlying layers to provide accurate and efficient data delivery
- Typical protocols: <u>FTP</u> (File Transfer Protocol) -For file transfer

  <u>Telnet</u> – Remote terminal protocol-For remote login on any other computer on the network

  <u>SMTP</u> (Simple Mail Transfer Protocol)-For mail transfer

  <u>HTTP (</u>Hypertext Transfer Protocol)-For Web browsing
- Encompasses same functions as these OSI Model layers-Application, Presentation and Session

<u>Transport Layer</u>

- Offering a reliable byte-stream delivery service
- Functions the same as the Transport layer in OSI
- Synchronize source and destination computers to set up the session between the respective computers

TCP is a connection-oriented protocol

Does not mean it has a physical connection between sender and receiver

TCP provides the function to allow a connection virtually exists – also called virtual circuit

UDP provides the functions:

Dividing a chunk of data into segments

Reassembly segments into the original chunk

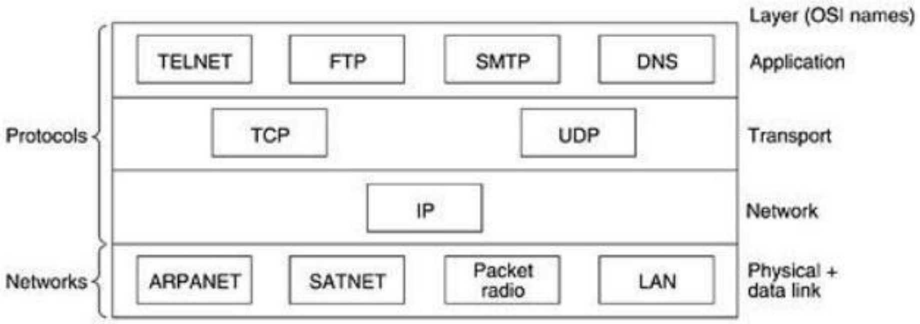
Provide further the functions such as reordering and data resend

<u>Internet Layer</u>

- The network layer, also called the internet layer, deals with packets and connects independent networks to transport the packets across network boundaries. The network layer protocols are the Internet Protocol (IP) and the Internet Control Message Protocol (ICMP), which is used for error reporting.

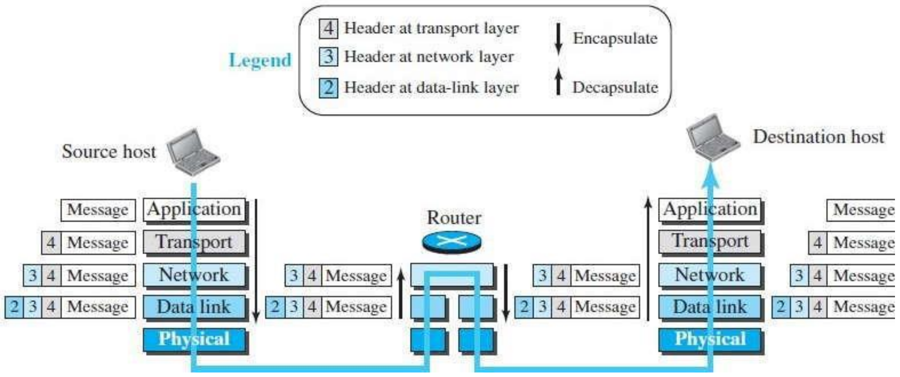
Host-to-network layer

- The **Host-to-network layer** is the lowest **layer** of the **TCP/IP** reference model. It combines the link **layer** and the physical **layer** of the ISO/OSI model. At this **layer**, data is transferred between adjacent **network** nodes in a WAN or between nodes on the same LAN.



Protocols and networks in the TCP/IP model

Encapsulation and Decapsulation



Encapsulation at the Source Host

At the source, we have only encapsulation.

1. At the application layer, the data to be exchanged is referred to as a *message*. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the message. The message is passed to the transport layer.

2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that want to communicate plus some more information that is needed for the end-toend delivery of the message, such as information needed for flow, error control, or congestion control. The result is the trans port-layer packet, which is called the *segment* (in TCP) and the *user datagram* (in UDP). The transport layer then passes the packet to the network layer.

3. The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for error checking of the header, fragmentation information, and so on. The result is the network-layer packet, called a *datagram*. The network layer then passes the packet to the data-link layer.

4. The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a *frame*. The frame is passed to the physical layer for transmission.

Decapsulation and Encapsulation at the Router

At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

1. After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.

2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.

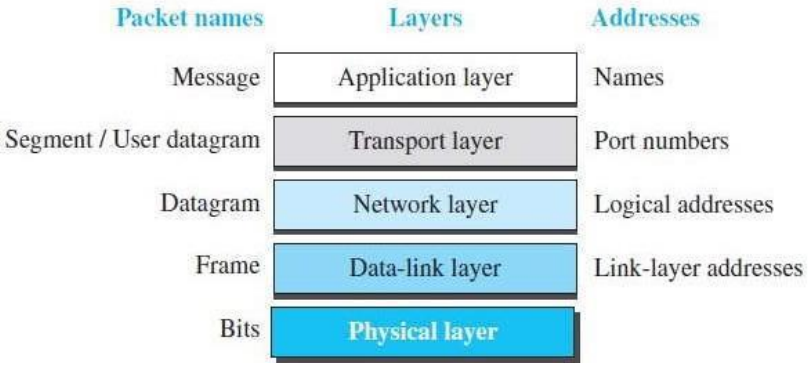
3. The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.

Decapsulation at the Destination Host

At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer. It is necessary to say that decapsulation in the host involves error checking.

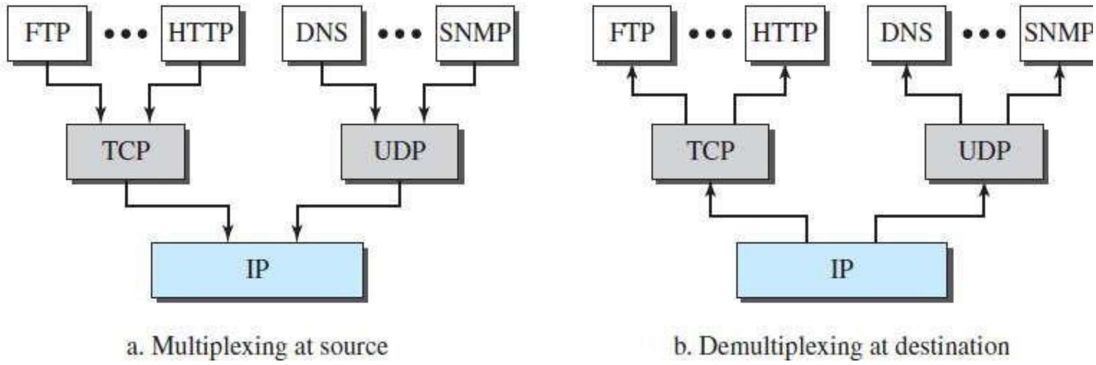
Addressing

It is worth mentioning another concept related to protocol layering in the Internet, *addressing*. As we discussed before, we have logical communication between pairs of layers in this model. Any communication that involves two parties needs two addresses: source address and destination address. Although it looks as if we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address. Figure shows the addressing at each layer. As the figure shows, there is a relationship between the layer, the address used in that layer, and the packet name at that layer. At the application layer, we normally use names to define the site that provides services, such as *someorg.com*, or the e-mailaddress, such as *somebody@coldmail.com.* At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination. Port numbers are local addresses that distinguish between several programs running at the same time. At the network-layer, the addresses are global, with the whole Internet as the scope. A network-layer address uniquely defines the connection of a device to the Internet. The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN). We will come back to these addresses in future chapters.



Multiplexing and Demultiplexing

Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination. Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time); demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time). Figure shows the concept of multiplexing and demultiplexing at the three upper layers.



a. Multiplexing at source

b. Demultiplexing at destination

To be able to multiplex and demultiplex, a protocol needs to have a field in its header to identify to which protocol the encapsulated packets belong. At the transport layer, either UDP or TCP can accept a message from several application-layer protocols. At the network layer, IP can accept a segment from TCP or a user datagram from UDP. IP can also accept a packet from other protocols such as ICMP, IGMP, and so on. At the data-link layer, a frame may carry the payload coming from IP or other protocols such as ARP.

Introduction to Physical layer

Physical layer in the OSI model plays the role of interacting with actual hardware and signaling mechanism. Physical layer is the only layer of OSI network model which actually deals with the physical connectivity of two different stations. This layer defines the hardware equipment, cabling, wiring, frequencies, pulses used to represent binary signals etc.

Physical layer provides its services to Data-link layer. Data-link layer hands over frames to physical layer. Physical layer converts them to electrical pulses, which represent binary data.The binary data is then sent over the wired or wireless media.

Signals

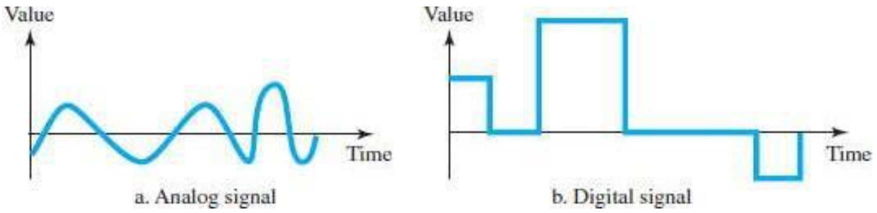
Analog and Digital Signals: Like the data they represent, **signals** can be either analog or digital.

An **analog signal** has infinitely many levels of intensity (*meaning strength/power*) over a period of time. As the wave moves from value *A* to value *B*, it passes through and includes an infinite number of values along its path.

A **digital signal,** on the other hand, can have only a limited number of defined values. Although each value can be any number, it is often as simple as 1 and 0.

The simplest way to show signals is by plotting them on a pair of perpendicular axes. The vertical axis represents the value or strength of a signal. The horizontal axis represents time.

Figure 1.29 illustrates an analog signal and a digital signal. The curve representing the analog signal passes through an infinite number of points. The vertical lines of the digital signal, however, demonstrate the sudden jump that the signal makes from value to



a. Analog signal

b. Digital signal

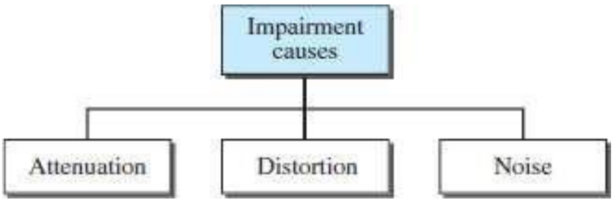
Comparison Of Analog And Digital Signals

Periodic and Non-periodic: Both analog and digital signals can take one of two forms: *Periodic* or *Non-periodic* A **periodic signal** completes a pattern within a measurable time frame, called a **period,** and repeats that pattern over subsequent identical periods. The completion of one full pattern is called a **cycle.** A **non-periodic signal** changes without exhibiting a pattern or cycle that repeats over time. Both analog and digital signals can be periodic or non-periodic.

In data communications, we commonly use periodic analog signals and non-periodic digital signals.

TRANSMISSION IMPAIRMENT:

Signals travel through transmission media, which are not perfect. The imperfection causes signal impairment. This means that the signal at the beginning of the medium is not the same as the signal at the end of the medium. What is sent is not what is received. Three causes of impairment are ***attenuation***, ***distortion***, and ***noise.***



Attenuation means a loss of energy. When a signal, simple or composite, travels through a medium, it loses some of its energy in overcoming the resistance of the medium. That is why a wire carrying electric signals gets warm, if not hot, after a while. Some of the electrical energy in the signal is converted to heat. To compensate for this loss, amplifiers are used to amplify (*meaning enlarge on/go into detail/develop/expand/clarify/add details to*) the signal.

Decibel: To show that a signal has lost or gained strength, engineers use the unit of the decibel. The decibel (dB) measures the relative strengths of two signals or one signal at two different points. Note that the decibel is negative if a signal is attenuated and positive if a signal is amplified.

Distortion means that the signal changes its form or shape. Distortion can occur in a composite signal made of different frequencies. Each signal component has its own propagation speed through a medium and, therefore, its own delay in arriving at the final destination. Differences in delay may create a difference in phase if the delay is not exactly the same as the period duration.

Noise is another cause of impairment. Several types of noise, such as ***thermal noise***, ***induced noise***, ***crosstalk***, and ***impulse noise***, may corrupt the signal.

Thermal noise is the random motion of electrons in a wire, which creates an extra signal not originally sent by the transmitter.

Induced noise comes from sources such as motors and appliances. These devices act as a sending antenna, and the transmission medium acts as the receiving antenna.

Crosstalk is the effect of one wire on the other. One wire acts as a sending antenna and the other as the receiving antenna.

Impulse noise is a spike (a signal with high energy in a very short time) that comes from power lines, lightning, and so on.

DATA RATE LIMITS:

A very important consideration in data communications is how fast we can send data, in bits per second, over a channel. Data rate depends on three factors:

1. The bandwidth available

2. The level of the signals we use

3. The quality of the channel (the level of noise)

Two theoretical formulas were developed to calculate the data rate: one by ***Nyquist for a noiseless channel***, another by ***Shannon for a noisy channel***.

Noiseless Channel: Nyquist Bit Rate:

For a noiseless channel, the **Nyquist bit rate** formula defines the theoretical maximum bit rate **BitRate = 2 x bandwidth x log2L**

In this formula, bandwidth is the bandwidth of the channel, *L* is the number of signal levels used to represent data, and BitRate is the bit rate in bits per second. According to the formula, we might think that, given a specific bandwidth, we can have any bit rate we want by increasing the number of signal levels.

Although the idea is theoretically correct, practically there is a limit. When we increase the number of signal levels, we impose a burden on the receiver.

Increasing the levels of a signal may reduce the reliability of the system.

Noisy Channel: Shannon Capacity:

In reality, we cannot have a noiseless channel; the channel is always noisy. In 1944, Claude Shannon introduced a formula, called the **Shannon capacity,** to determine the theoretical highest data rate for a noisy channel: **Capacity = bandwidth x log2 (1 + SNR)**

In this formula, bandwidth is the bandwidth of the channel, SNR is the signal-to-noise ratio, and capacity is the capacity of the channel in bits per second.

The Shannon capacity gives us the upper limit; the Nyquist formula tells us how many signal levels we need.

PERFORMANCE:

One important issue in networking is the performance of the network—how good is it? There are certain characteristics that measure the network performance which are given as follows:

Bandwidth

One characteristic that measures network performance is bandwidth. However, the term can be used in two different contexts with two different measuring values: bandwidth in hertz and bandwidth in bits per second.

Bandwidth in Hertz: Bandwidth in hertz is the range of frequencies contained in a composite signal or the range of frequencies a channel can pass. For example, we can say the bandwidth of a subscriber telephone line is 4 kHz.

Bandwidth in Bits per Seconds: The term *bandwidth* can also refer to the number of bits per second that a channel, a link, or even a network can transmit. For example, one can say the bandwidth of a Fast Ethernet network (or the links in this network) is a maximum of 100 Mbps. This means that this network can send 100 Mbps.

Relationship: There is an explicit relationship between the bandwidth in hertz and bandwidth in bits per second. Basically, an increase in bandwidth in hertz means an increase in bandwidth in bits per second.

Throughput:

The **throughput** is a measure of how fast we can actually send data through a network. Although, at first glance, bandwidth in bits per second and throughput seem the same, they are different. A link may have a bandwidth of *B* bps, but we can only send *T* bps through this link with *T* always less than *B*.

For example, we may have a link with a bandwidth of 1 Mbps, but the devices connected to the end of the link may handle only 200 kbps. This means that we cannot send more than 200 kbps through this link.

Imagine a highway designed to transmit 1000 cars per minute from one point to another. However, if there is congestion on the road, this figure may be reduced to 100 cars per minute. The bandwidth is 1000 cars per minute; the throughput is 100 cars per minute.

Latency (Delay):

The **latency** or delay defines how long it takes for an entire message to completely arrive at the destination from the time the first bit is sent out from the source. We can say that latency is made of four components: ***propagation time***, ***transmission time***, ***queuing time*** and ***processing delay***.

Latency = **propagation time + transmission time + queuing time + processing delay**

Propagation Time: **Propagation time** measures the time required for a bit to travel from the source to the destination. The propagation time is calculated by dividing the distance by the propagation speed.

Propagation time = Distance / (Propagation Speed)

Transmission Time: In data communications we don’t send just 1 bit, we send a message. The first bit may take a time equal to the propagation time to reach its destination; the last bit also may take the same amount of time. However, there is a time between the first bit leaving the sender and the last bit arriving at the receiver.

The first bit leaves earlier and arrives earlier; the last bit leaves later and arrives later. The **transmission time**
of a message depends on the size of the message and the bandwidth of the channel.

Transmission time = (Message size) / Bandwidth

Queuing Time: The third component in latency is the **queuing time**, the time needed for each intermediate or end device to hold the message before it can be processed. The queuing time is not a fixed factor; it changes with the load imposed on the network.

When there is heavy traffic on the network, the queuing time increases. An intermediate device, such as a router, queues they arrived messages and processes them one by one. If there are many messages, each message will have to wait.

Bandwidth-Delay Product

Bandwidth and delay are two performance metrics of a link. ***The bandwidth-delay product defines the number of bits that can fill the link.***

JITTER:

Another performance issue that is related to delay is **jitter.** We can roughly say that jitter is a problem if different packets of data encounter different delays and the application using the data at the receiver site is time-sensitive (audio and video data, for example). If the delay for the first packet is 20 ms, for the second is 45 ms, and for the third is 40 ms, then the real-time application that uses the packets endures jitter.

TRANSMISSION MEDIA

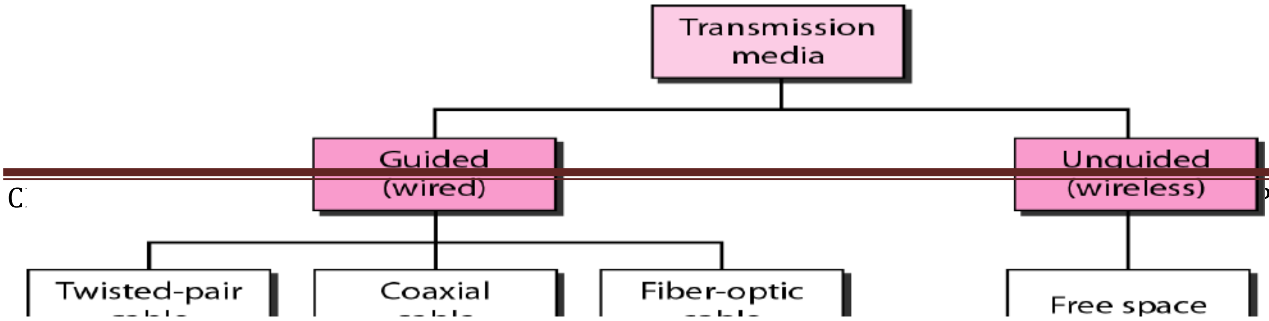
The media over which the information between two computer systems is sent, called transmission media. Transmission media comes in two forms.

- **Guided Media**

All communication wires/cables are guided media, such as UTP, coaxial cables, and fiber Optics. In this media, the sender and receiver are directly connected and the information is send (guided) through it.

- **Unguided Media**

Wireless or open air space is said to be unguided media, because there is no connectivity between the sender and receiver. Information is spread over the air, and anyone including the actual recipient may collect the information.



Magnetic Media

One of the most convenient way to transfer data from one computer to another, even before the birth of networking, was to save it on some storage media and transfer physical from one station to another. Though it may seem old-fashion way in today’s world of high speed internet, but when the size of data is huge, the magnetic media comes into play.

For example, a bank has to handle and transfer huge data of its customer, which stores a backup of it at some geographically far-away place for security reasons and to keep it from uncertain calamities. If the bank needs to store its huge backup data then its, transfer through internet is not feasible. The WAN links may not support such high speed. Even if they do; the cost too high to afford.
In these cases, data backup is stored onto magnetic tapes or magnetic discs, and then shifted physically at remote places.

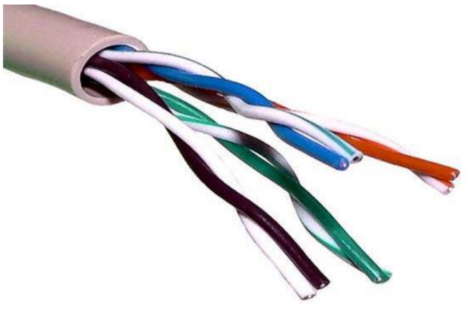
Guided Media

Twisted Pair Cable

A twisted pair cable is made of two plastic insulated copper wires twisted together to form a single media. Out of these two wires, only one carries actual signal and another is used for ground reference. The twists between wires are helpful in reducing noise (electro-magnetic interference) and crosstalk.

There are two types of twisted pair cables:

- Shielded Twisted Pair (STP) Cable
- Unshielded Twisted Pair(UTP) Cable



STP cables comes with twisted wire pair covered in metal foil. This makes it more indifferent to noise and crosstalk.

UTP has seven categories, each suitable for specific use. In computer networks, Cat-5, Cat-5e, and Cat-6 cables are mostly used. UTP cables are connected by RJ45 connectors.

Coaxial Cable

Coaxial cable has two wires of copper. The core wire lies in the center and it is made of solid conductor. The core is enclosed in an insulating sheath. The second wire is wrapped around over the sheath and that too in turn encased by insulator sheath. This all is covered by plastic cover.

Because of its structure, the coaxial cable is capable of carrying high frequency signals than that of twisted pair cable. The wrapped structure provides it a good shield against noise and cross talk. Coaxial cables provide high bandwidth rates of up to 450 mbps.

There are three categories of coaxial cables namely, RG-59 (Cable TV), RG-58 (Thin Ethernet), and RG-11 (Thick Ethernet). RG stands for Radio Government.
Cables are connected using BNC connector and BNC-T. BNC terminator is used to terminate the wire at the far ends.

Power Lines

Power Line communication (PLC) is Layer-1 (Physical Layer) technology which uses power cables to transmit data signals.In PLC, modulated data is sent over the cables. The receiver on the other end de-modulates and interprets the data.

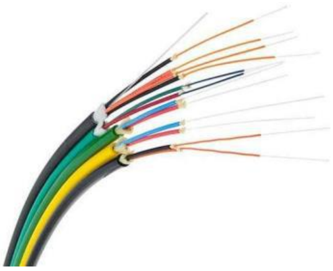
Because power lines are widely deployed, PLC can make all powered devices controlled and monitored. PLC works in half-duplex. There are two types of PLC:Narrow band PLC and Broad band PLC

Narrow band PLC provides lower data rates up to 100s of kbps, as they work at lower frequencies (3-5000 kHz).They can be spread over several kilometers.

Broadband PLC provides higher data rates up to 100s of Mbps and works at higher frequencies (1.8 – 250 MHz).They cannot be as much extended as Narrowband PLC.

Fiber Optics

Fiber Optic works on the properties of light. When light ray hits at critical angle it tends to refracts at 90 degree. This property has been used in fiber optic. The core of fiber optic cable is made of high quality glass or plastic. From one end of it light is emitted, it travels through it and at the other end light detector detects light stream and converts it to electric data.

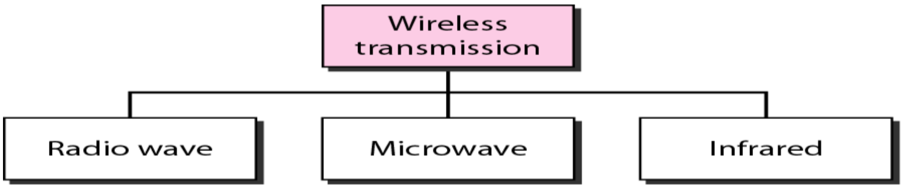


Fiber Optic provides the highest mode of speed. It comes in two modes, one is single mode fiber and second is multimode fiber. Single mode fiber can carry a single ray of light whereas multimode is capable of carrying multiple beams of light.

Fiber Optic also comes in unidirectional and bidirectional capabilities. To connect and access fiber optic special type of connectors are used. These can be Subscriber Channel (SC), Straight Tip (ST), or MT -RJ.

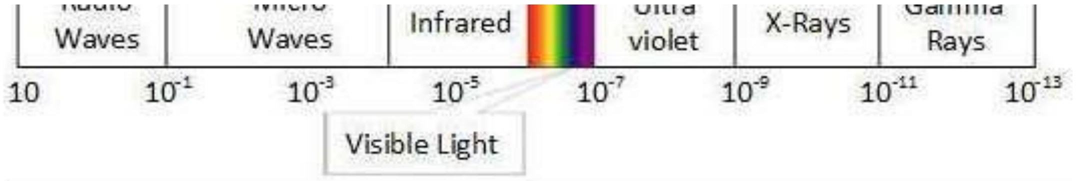
Unguided Media

Wireless transmission is a form of unguided media. Wireless communication involves no physical link established between two or more devices, communicating wirelessly. Wireless signals are spread over in the air and are received and interpreted by appropriate antennas.



When an antenna is attached to electrical circuit of a computer or wireless device, it converts the digital data into wireless signals and spread all over within its frequency range. The receptor on the other end receives these signals and converts them back to digital data.

A little part of electromagnetic spectrum can be used for wireless transmission.

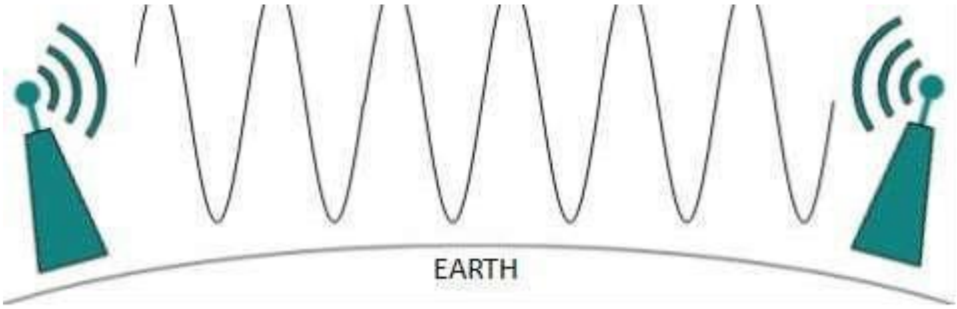


Radio Transmission

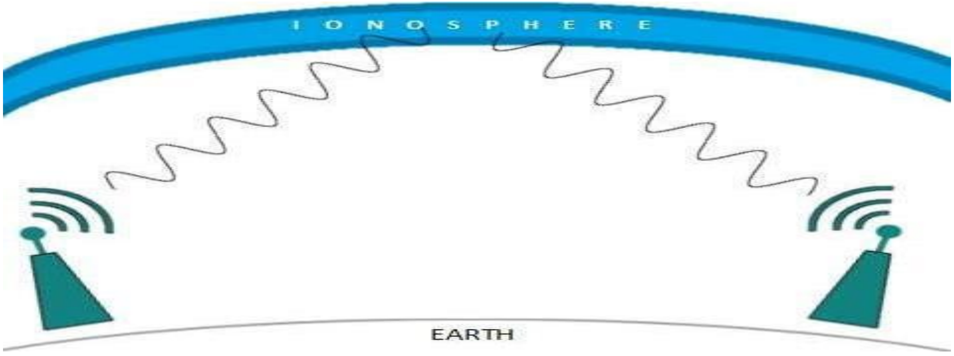
Radio frequency is easier to generate and because of its large wavelength it can penetrate through walls and structures alike. Radio waves can have wavelength from 1 mm – 100,000 km and have frequency ranging from 3 Hz (Extremely Low Frequency) to 300 GHz (Extremely High Frequency). Radio frequencies are sub-divided into six bands.

Radio waves at lower frequencies can travel through walls whereas higher RF can travel in straight line and bounce back. The power of low frequency waves decreases sharply as they cover long distance. High frequency radio waves have more power.

Lower frequencies such as VLF, LF, MF bands can travel on the ground up to 1000 kilometers, over the earth's surface.



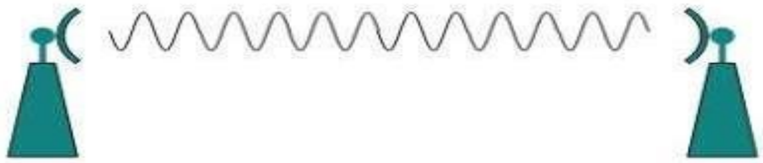
Radio waves of high frequencies are prone to be absorbed by rain and other obstacles. They use Ionosphere of earth atmosphere. High frequency radio waves such as HF and VHF bands are spread upwards. When they reach Ionosphere, they are refracted back to the earth.



Microwave Transmission

Electromagnetic waves above 100 MHz tend to travel in a straight line and signals over them can be sent by beaming those waves towards one particular station. Because Microwaves travels in straight lines, both sender and receiver must be aligned to be strictly in line-of-sight.

Microwaves can have wavelength ranging from 1 mm – 1 meter and frequency ranging from 300 MHz to 300 GHz.



Microwave antennas concentrate the waves making a beam of it. As shown in picture above, multiple antennas can be aligned to reach farther. Microwaves have higher frequencies and do not penetrate wall like obstacles.Microwave transmission depends highly upon the weather conditions and the frequency it is using.

Infrared Transmission

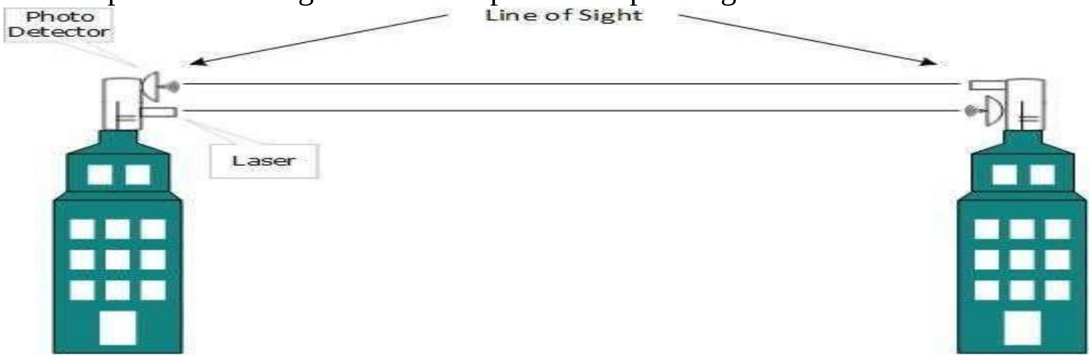
Infrared wave lies in between visible light spectrum and microwaves. It has wavelength of 700-nm to 1-mm and frequency ranges from 300-GHz to 430-THz.

Infrared wave is used for very short range communication purposes such as television and it's remote. Infrared travels in a straight line hence it is directional by nature. Because of high frequency range, Infrared cannot cross wall-like obstacles.

Light Transmission

Highest most electromagnetic spectrum which can be used for data transmission is light or optical signaling. This is achieved by means of LASER.

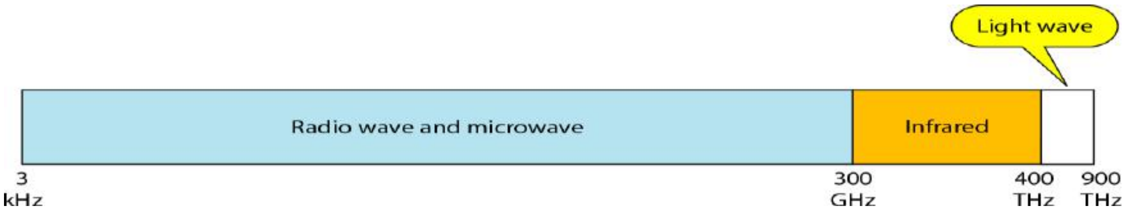
Because of frequency light uses, it tends to travel strictly in straight line. Hence the sender and receiver must be in the line-of-sight. Because laser transmission is unidirectional, at both ends of communication the laser and the photo-detector need to be installed. Laser beam is generally 1mm wide hence it is a work of precision to align two far receptors each pointing to lasers source.



Laser works as Tx (transmitter) and photo-detectors works as Rx (receiver).

Lasers cannot penetrate obstacles such as walls, rain, and thick fog. Additionally, laser beam is distorted by wind, atmosphere temperature, or variation in temperature in the path.

Laser is safe for data transmission as it is very difficult to tap 1mm wide laser without interrupting the communication channel.



UNIT 2
INTRODUCTION TO DATALINK LAYER

❖ INTRODUCTION:

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Figure shows the same scenario. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

COMMUNICATION AT THE DATA-LINK LAYER

SERVICES:

The data-link layer is located between the physical and the network layers. The data link layer provides services to the network layer; it receives services from the physical layer. The duty scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path. For this purpose, the data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame.

FRAMING: Definitely, the first service provided by the data-link layer is **framing**. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a **frame** before sending it to the next node. The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame.

FLOW CONTROL: The sending data-link layer at the end of a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer. If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed). Definitely, we cannot have an unlimited buffer size at the receiving side. We have two choices. The first choice is to let the receiving data-link layer drop the frames if its buffer is full. The second choice is to let the receiving data-link layer send a feedback to the sending data-link layer to ask it to stop or slow down. Different data-link-layer protocols use different strategies for flow control.

ERROR CONTROL: At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame. Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node.

CONGESTION CONTROL: Although a link may be congested with frames, which may result in frame loss, most data-link-layer protocols do not directly use a congestion control to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature.

TWO CATEGORIES OF LINKS: Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium; we can also have a data- link layer that uses only part of the capacity of the link. In other words, we can have a *point-to-point link* or a *broadcast link*. In a point-to-point link, the link is dedicated to the two devices; in a broadcast link, the link is shared between several pairs of devices.

Two Sub layers: To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sub layers: **data link control (DLC)** and **media access control (MAC).** The data link control sub layer deals with all issues common to both point-to-point and broadcast links; the media access control sub layer deals only with issues specific to broadcast links.

❖ LINK-LAYER ADDRESSING:

A *link-layer address* is sometimes called a *link address*, sometimes a *physical address*, and sometimes a *MAC address*.

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another. Figure demonstrates the concept in a small internet.

In the internet in Figure, we have three links and two routers. We also have shown only two hosts: Alice (source) and Bob (destination). For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L).

Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown three frames, one in each link. Each frame carries the same datagram with the same source and destination addresses (**N1** and **N8**), but the link-layer addresses of the frame change from link to link.

In link 1, the link-layer addresses are L1 and L2. In link 2, they are L4 and L5. In link 3, they are L7 and L8.

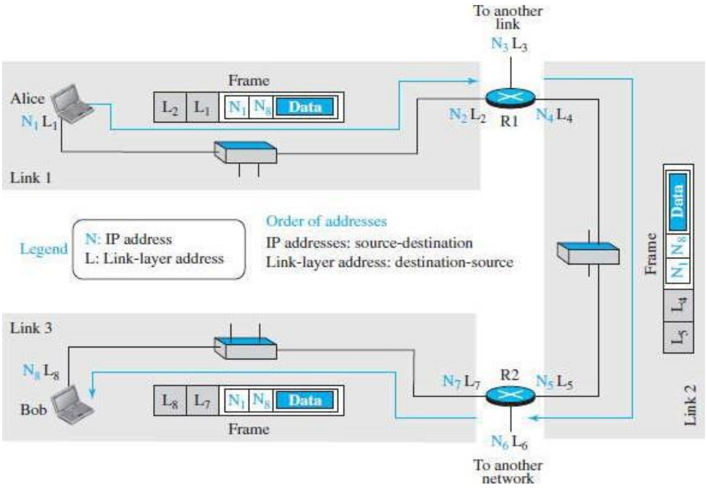


FIGURE: IP ADDRESSES AND LINK-LAYER ADDRESSES IN A SMALL INTERNET

Note that the IP addresses and the link-layer addresses are not in the same order. For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source.

Address Resolution Protocol (ARP):

Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router.

Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host. However, the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the **Address Resolution Protocol (ARP)** becomes helpful. ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

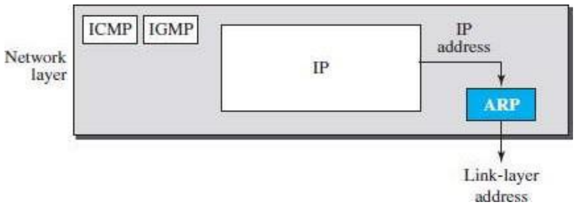


FIGURE: POSITION OF ARP IN TCP/IP PROTOCOL SUITE

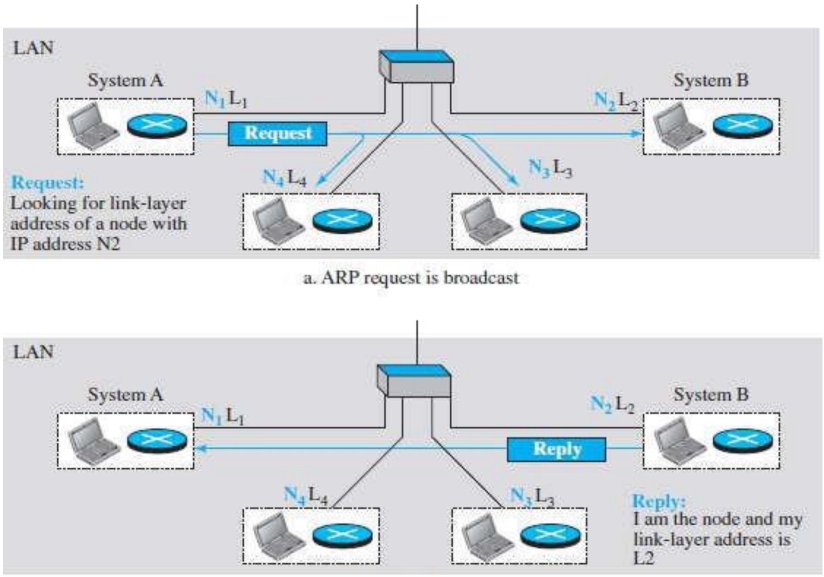
Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address.

Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient’s IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.

In Figure (a), the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address **N2**. System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient.

It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of **N2**. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure (b).

System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.



a. ARP request is broadcast

b. ARP reply is unicast

FIGURE: ARP OPERATION

Packet Format:

Figure shows the format of an ARP packet. The names of the fields are self-explanatory. The *hardware type* field defines the type of the link-layer protocol; Ethernet is given the type 1.

The *protocol type* field defines the network-layer protocol: IPv4 protocol is (0800)16. The source hardware and source protocol addresses are variable-length fields defining the link-layer and network- layer addresses of the sender.

The destination hardware address and destination protocol address fields define the receiver link- layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

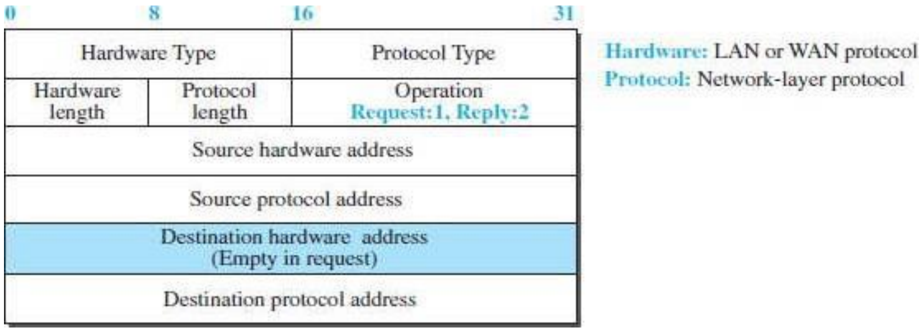


FIGURE: ARP PACKET

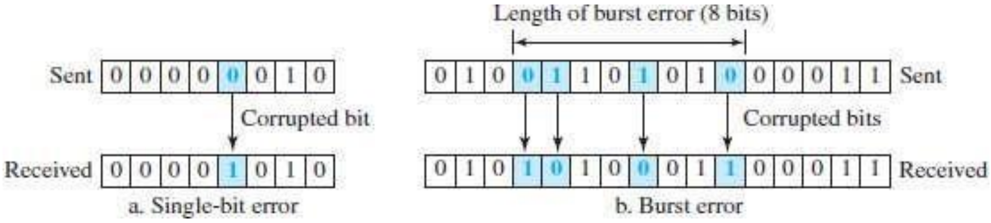
ERROR DETECTION AND CORRECTION

❖ **Types of Errors:**

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference.** This interference can change the shape of the signal. The term ***single-bit error*** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

The term ***burst error*** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1. Figure 2.8 shows the effect of a single -bit and a burst error on a data unit.

FIGURE: SINGLE-BIT AND BURST ERROR



Redundancy:

The central concept in detecting or correcting errors is **redundancy**. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

Detection versus Correction:

The correction of errors is more difficult than the detection. In **error detection**, we are only looking to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of corrupted bits. A single-bit error is the same for us as a burst error.

In **error correction**, we need to know the exact number of bits that are corrupted and, more importantly, their location in the message. The number of errors and the size of the message are important factors.

If we need to correct a single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 (permutation of 8 by 2) possibilities. You can imagine the receiver’s difficulty in finding 10 errors in a data unit of 1000 bits.

- **Error Detection:**

Cyclic Redundancy Check:

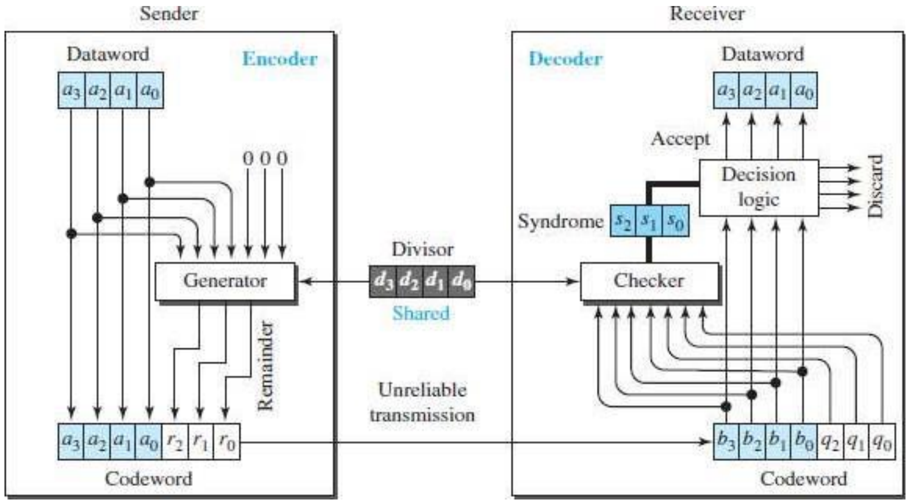


FIGURE: CRC ENCODER AND DECODER

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

Decoder: The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error with a high probability; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded.

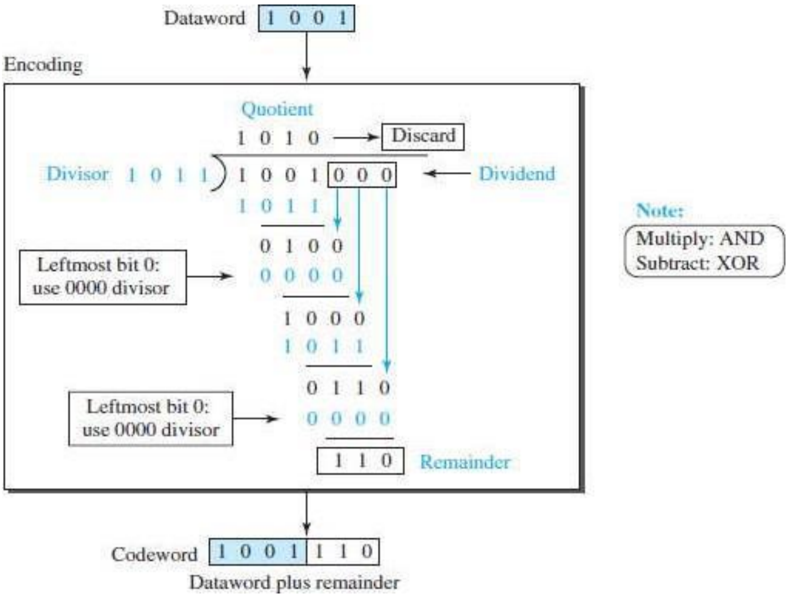


FIGURE: DIVISION IN CRC ENCODER

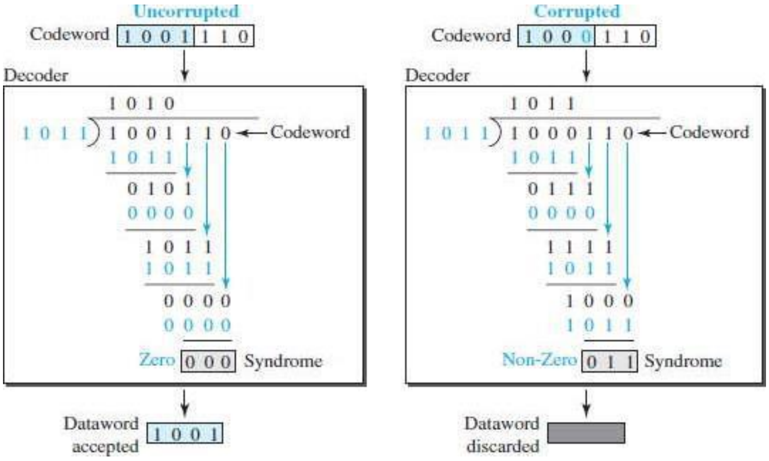


FIGURE: DIVISION IN THE CRC DECODER FOR TWO CASES

The above Figure shows two cases: The left-hand figure shows the value of the syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is a single error. The syndrome is not all 0s (it is 011).

ADVANTAGES OF CYCLIC CODES:

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware. This has made cyclic codes a good candidate for many networks.

CHECKSUM:

Checksum is an error-detecting technique that can be applied to a message of any length. In the Internet, the checksum technique is mostly used at the network and transport layer rather than the data- link layer.

At the source, the message is first divided into *m*-bit units. The generator then creates an extra *m*- bit unit called the ***checksum,*** which is sent with the message. At the destination, the checker creates a new checksum from the combination of the message and sent checksum. If the new checksum is all 0s, the message is accepted; otherwise, the message is discarded (Figure). Note that in the real implementation, the checksum unit is not necessarily added at the end of the message; it can be inserted in the middle of the message.

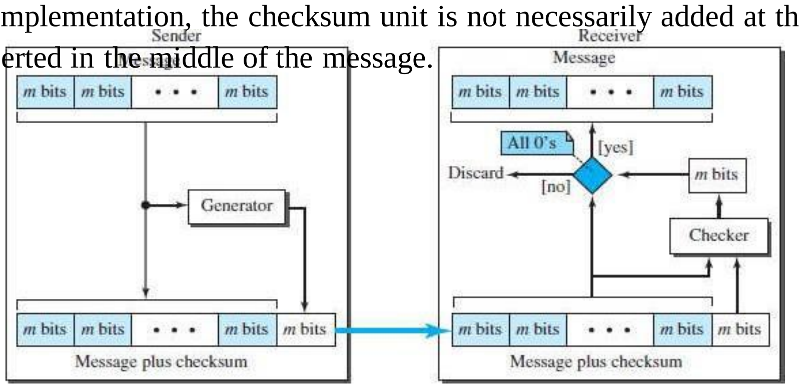


FIGURE: CHECKSUM

Suppose the message is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers.

The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the message is not accepted.

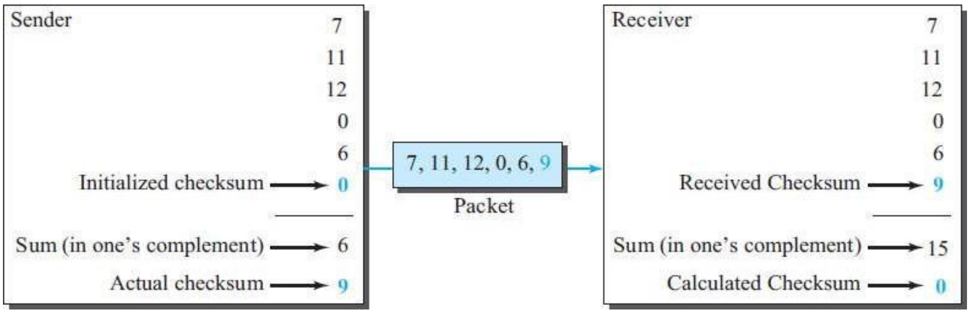
One's Complement Addition:

The previous example has one major drawback. Each number can be written as a 4-bit word (each is less than 15) except for the sum. One solution is to use **one's complement** arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^m - 1$ using only m bits. If the number has more than m bits, the extra leftmost bits need to be added to the m rightmost bits (wrapping).

In the previous example, the decimal number 36 in binary is (100100)2. To change it to a 4-bit number we add the extra leftmost bit to the right four bits as shown below.

$$\mathbf{(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}}$$

Instead of sending 36 as the sum, we can send 6 as the sum (7, 11, 12, 0, 6, **6**). The receiver can add the first five numbers in one's complement arithmetic. If the result is 6, the numbers are accepted; otherwise, they are rejected.



❖ <u>FORWARD ERROR CORRECTION:</u>

We need to correct the error or reproduce the packet immediately. Several schemes have been designed and used in this case that is collectively referred to as **forward error correction** (**FEC**) techniques.

HAMMING DISTANCE:

To detect s errors, the minimum Hamming distance should be $d$min = s + 1. For error detection, we definitely need more distance. It can be shown that to detect t errors, we need to have $d$min = 2t + 1. In other words, if we want to correct 10 bits in a packet, we need to make the minimum hamming distance 21 bits, which means a lot of redundant bits, need to be sent with the data.

To give an example, consider the famous BCH code. In this code, if data is 99 bits, we need to send 255 bits (extra 156 bits) to correct just 23 possible bit errors. Most of the time we cannot afford such a redundancy.

CHUNK INTERLEAVING: Another way to achieve FEC in multimedia is to allow some small chunks to be missing at the receiver. We cannot afford to let all the chunks belonging to the same packet be missing; however, we can afford to let one chunk be missing in each packet. Figure shows that we can divide each packet into 5 chunks (normally the number is much larger).

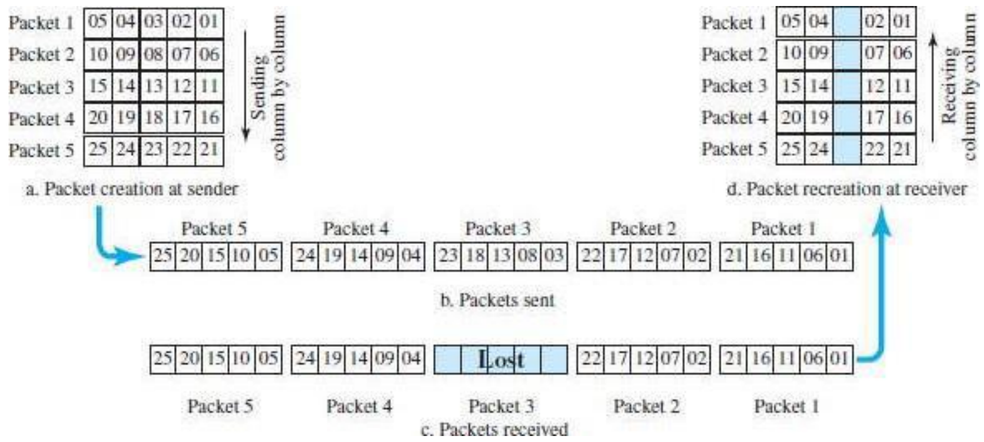


FIGURE: INTERLEAVING

We can then create data chunk by chunk (horizontally), but combine the chunks into packets vertically. In this case, each packet sent carries a chunk from several original packets. If the packet is lost, we miss only one chunk in each packet, which is normally acceptable in multimedia communication.

COMBINING HAMMING DISTANCE AND INTERLEAVING:

Hamming distance and interleaving can be combined. We can first create *n*-bit packets that can correct *t*- bit errors. Then we interleave *m* rows and send the bits column by column. In this way, we can automatically correct burst errors up to $m \times t$-bit errors.

COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS:

Still another solution is to create a duplicate of each packet with a low-resolution redundancy and combine the redundant version with the next packet. For example, we can create four low-resolution packets out of five high-resolution packets and send them as shown in Figure. If a packet is lost, we can use the low-resolution version from the next packet. Note that the low-resolution section in the first packet is empty.

In this method, if the last packet is lost, it cannot be recovered, but we use the low-resolution version of a packet if the lost packet is not the last one. The audio and video reproduction does not have the same quality, but the lack of quality is not recognized most of the time.

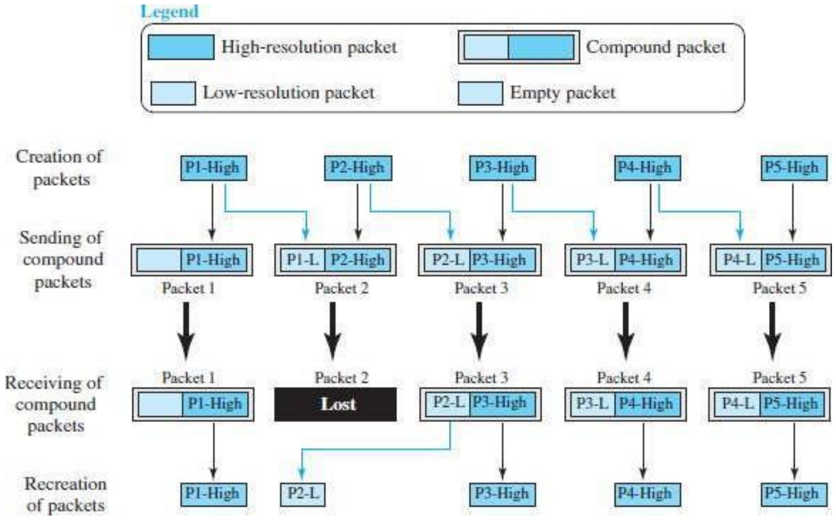


FIGURE: COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS

DATA LINK CONTROL

❖ DLC SERVICES:

The **data link control (DLC)** deals with procedures for communication between two adjacent nodes— node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include *framing* and *flow and error control*.

FRAMING: The data-link layer, needs to pack bits into frames, so that each frame is distinguishable from another. *Framing* in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Although the whole message could be packed in one frame, which is not normally done; one reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole frame. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Character-Oriented Framing:

To separate one frame from the next, an 8-bit (1-byte) **flag** is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure 2.17 shows the format of a frame in a character -oriented protocol.

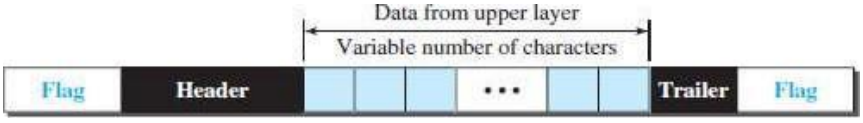


FIGURE: A FRAME IN A CHARACTER-ORIENTED PROTOCOL

Byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the *escape character (ESC)* and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag. Figure shows the situation.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag? To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

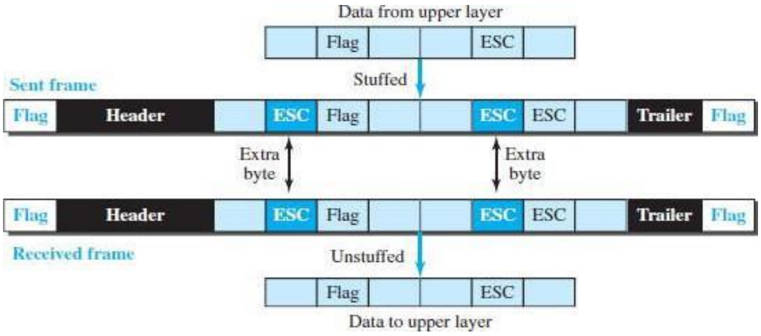


FIGURE: BYTE STUFFING AND UNSTUFFING

Bit-Oriented Framing:

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver. This means that if the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken for a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

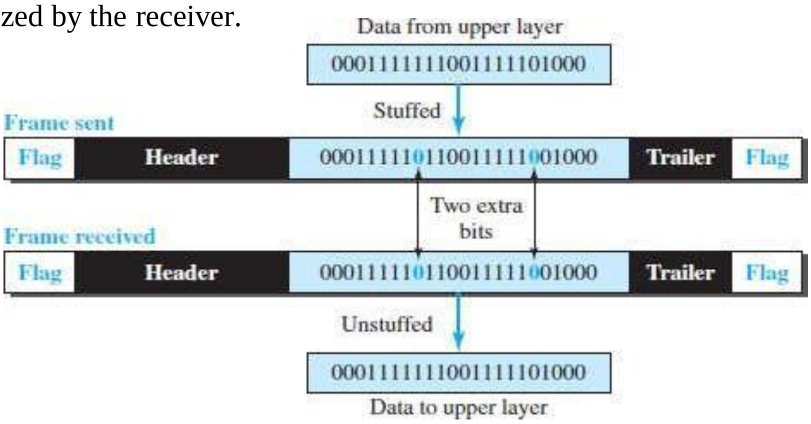


FIGURE: BIT STUFFING AND UNSTUFFING

FLOW AND ERROR CONTROL:

If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.

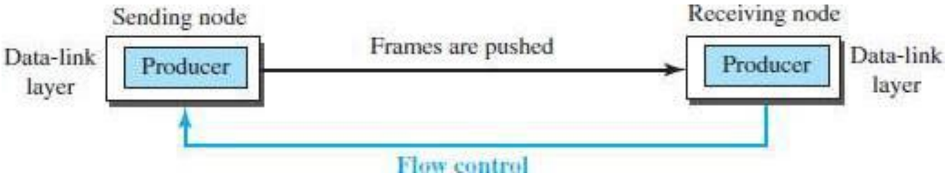


FIGURE: FLOW CONTROL AT THE DATA-LINK LAYER

Buffers: Although flow control can be implemented in several ways, one of the solutions is normally to use two *buffers*; one at the sending data-link layer and the other at the receiving data-link layer. A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control: Since the underlying technology at the physical layer is not fully reliable, we need to implement error control at the data-link layer to prevent the receiving node from delivering corrupted packets to its network layer.

Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

- In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.
- In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

❖ DATA-LINK LAYER PROTOCOLS:

SIMPLE PROTOCOL:

Our first protocol is a **simple protocol** with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives. In other words, the receiver can never be overwhelmed with incoming frames. Figure shows the layout for this protocol.

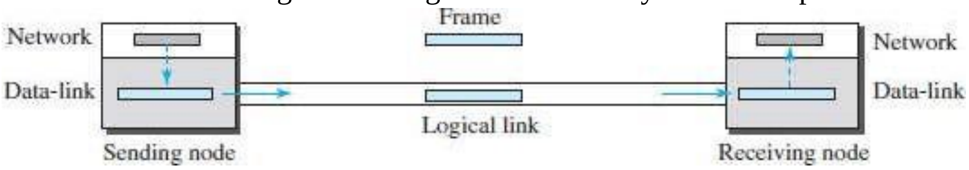
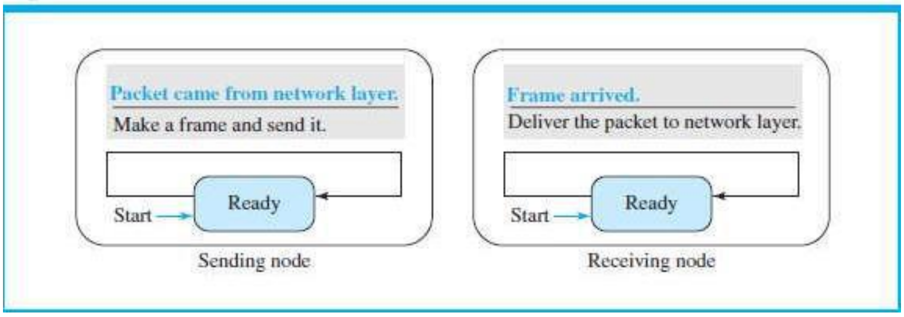


FIGURE: SIMPLE PROTOCOL

The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender and receiver provide transmission services for their network layers.


Packet came from network layer.
Make a frame and send it.
Start
Ready
Sending node
Frame arrived.
Deliver the packet to network layer.
Start
Ready
Receiving node

FSM OF SIMPLE PROTOCOL

STOP-AND-WAIT PROTOCOL:

Stop-and-Wait protocol uses both flow and error control. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame.

When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.

Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.

Figure shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

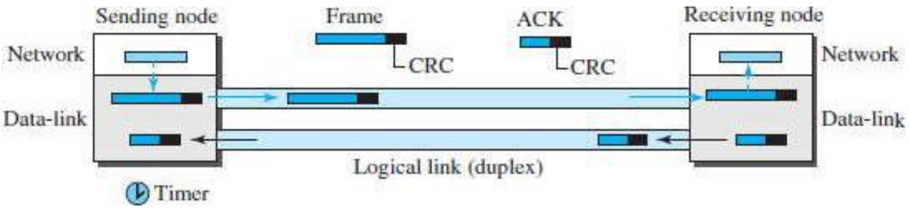


FIGURE: STOP-AND-WAIT PROTOCOL

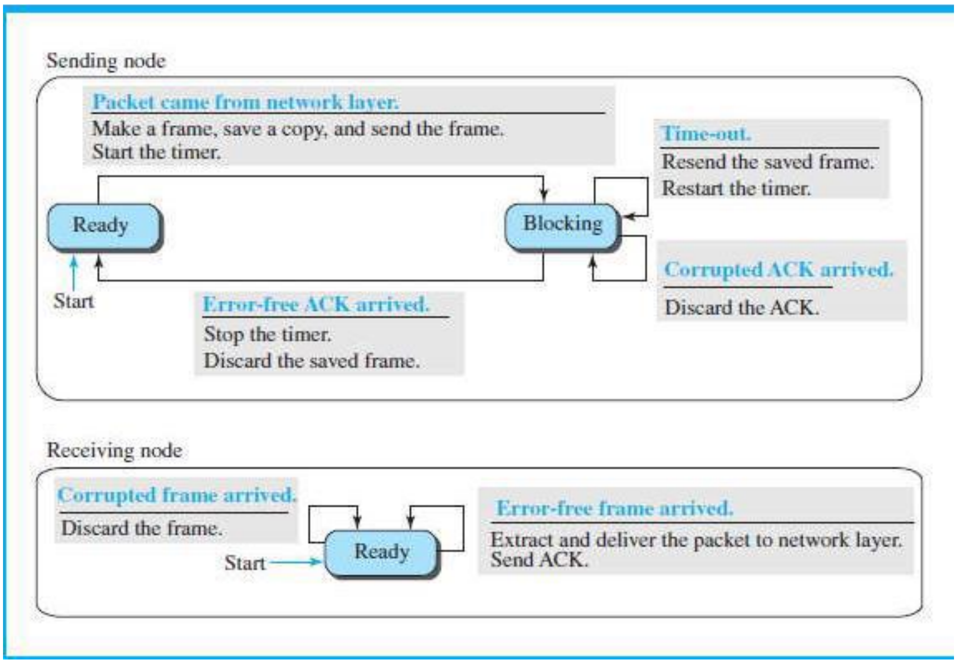


FIG: FSM OF STOP-AND-WAIT PROTOCOL

Piggybacking: The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction. Protocols have been designed in the past to allow data to flow in both directions. However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction. In other words, when node A is sending data to node B, Node A also acknowledges the data received from node B. Because piggybacking makes communication at the data link layer more complicated, it is not a common practice.

HDLC:

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

Configurations and Transfer Modes: HDLC provides two common transfer modes that can be used in different configurations: *normal response mode (NRM)* and *asynchronous balanced* *mode (ABM)*. In *normal response mode (NRM)*, the station configuration is unbalanced.

We have one primary station and multiple secondary stations. A *primary station* can send commands; a *secondary station* can only respond. The NRM is used for both point-to-point and multipoint links, as shown in Figure.

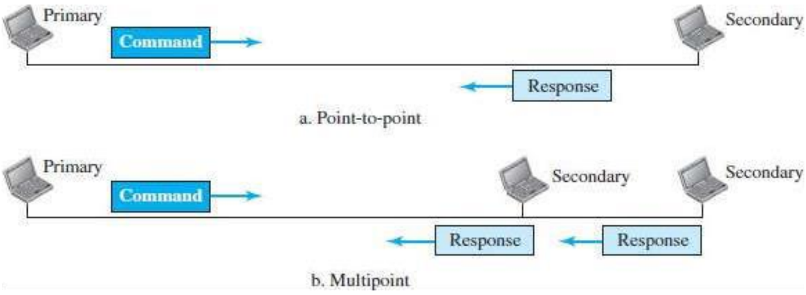


FIGURE: NORMAL RESPONSE MODE

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

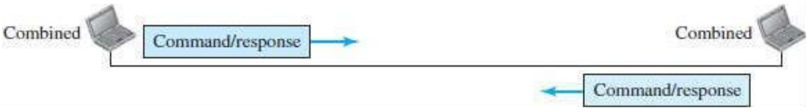


FIGURE: ASYNCHRONOUS BALANCED MODE

Framing: To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: *information frames (I-frames)*, *supervisory frames (S-frames)*, and *unnumbered frames (U-frames)*.

Each type of frame serves as an envelope for the transmission of a different type of message. I- frames are used to data-link user data and control information relating to user data (piggybacking).

S-frames are used only to transport control information. U-frames are reserved for system management. Information carried by U-frames is intended for managing the link itself. Each frame in HDLC may contain up to six fields, as shown in Figure: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple- frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

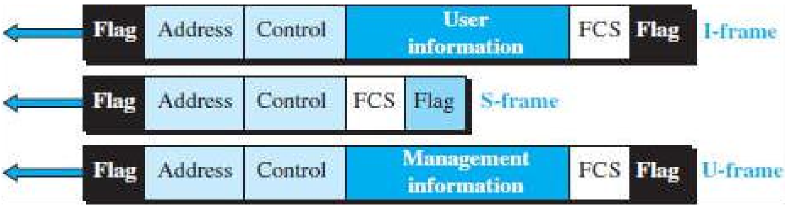


FIGURE 2.27: HDLC FRAMES

- ***Flag field.*** This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.
- ***Address field.*** This field contains the address of the secondary station. If a primary station created the frame, it contains *to* address. If a secondary station creates the frame, it contains *from* address. The address field can be one byte or several bytes long, depending on the needs of the network.
- ***Control field.*** The control field is one or two bytes used for flow and error control.
- ***Information field.*** The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- ***FCS field.*** The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte CRC.

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

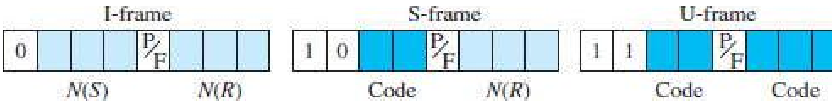


FIGURE: CONTROL FIELD FORMAT FOR THE DIFFERENT FRAME TYPES

Control Field for I-Frames

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

Control Field for S-Frames

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

- ***Receive ready (RR).*** If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value of the *N*(*R*) field defines the acknowledgment number.
- ***Receive not ready (RNR).*** If the value of the code subfield is 10, it is an RNR Sframe. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion-control mechanism by asking the sender to slow down. The value of *N*(*R*) is the acknowledgment number.
- ***Reject (REJ).*** If the value of the code subfield is 01, it is an REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-*N* ARQ to improve the efficiency of the process by informing the sender, before the sender timer expires, that the last frame is lost or damaged. The value of *N*(*R*) is the negative acknowledgment number.
- ***Selective reject (SREJ).*** If the value of the code subfield is 11, it is an SREJ Sframe. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat*. The value of *N*(*R*) is the negative acknowledgment number.

Control Field for U-Frames

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

POINT-TO-POINT PROTOCOL (PPP):

One of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP).** Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point- to-point protocol at the data-link layer. PPP is by far the most common.

Services: The designers of PPP have included several services to make it suitable for a point-to-point protocol, but have ignored some traditional services to make it simple.

Services Provided by PPP: PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data. PPP is designed to accept payloads from several network layers (not only IP).

Authentication is also provided in the protocol, but it is optional. The new version of PPP, called *Multilink PPP*, provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

Services Not Provided by PPP: PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver. PPP has a very simple mechanism for error control. A CRC field is used to detect errors.

If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

Framing:

PPP uses a character-oriented (or byte-oriented) frame. Figure shows the format of a PPP frame.

The description of each field follows:

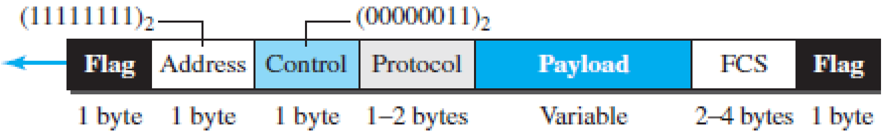


FIGURE: PPP FRAME FORMAT

- ***Flag.*** A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110.

- ***Address.*** The address field in this protocol is a constant value and set to 11111111 (broadcast address).

- ***Control.*** This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.

- ***Protocol.*** The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.

- ***Payload field.*** The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation.

  - The data field is byte-stuffed if the flag byte pattern appears in this field.

  - Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.

- ***FCS.*** The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

(i) Link Control Protocol:

The **Link Control Protocol (LCP)** is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established.

(ii)Authentication Protocols:

Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. *Authentication* means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP:

The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two- step process:

a. The user who wants to access a system sends authentication identification (usually the user name) and a password.

b. The system checks the validity of the identification and password and either accepts or denies connection.

CHAP:

The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way handshaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

a. The system sends the user a challenge packet containing a challenge value, usually a few bytes.

b. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.

c. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret.

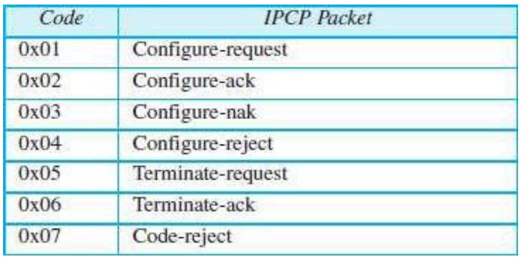
(iii) Network Control Protocols:

PPP is a multiple-network-layer protocol. It can carry a network-layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets.

IPCP:

One NCP protocol is the **Internet Protocol Control Protocol (IPCP).** This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 2.30. IPCP defines seven packets, distinguished by their code values, as shown in Table.

Other Protocols: There are other NCP protocols for other network-layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on.



Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject

TABLE 2.4: CODE VALUE FOR IPCP PACKETS

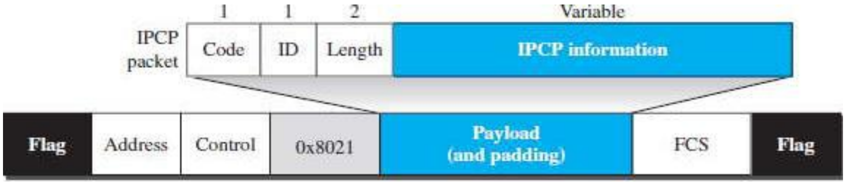


FIGURE 2.30: IPCP PACKET ENCAPSULATED IN PPP FRAME

Multilink PPP:

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure.

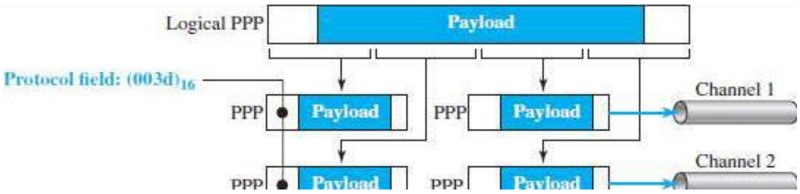


FIGURE: MULTILINK PPP

MEDIA ACCESS CONTROL (MAC)

When nodes or stations are connected and use a common link, called a *multipoint* or *broadcast link*, we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking in an assembly.

Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called *media access control (MAC).* We categorize them into three groups, as shown in Figure.

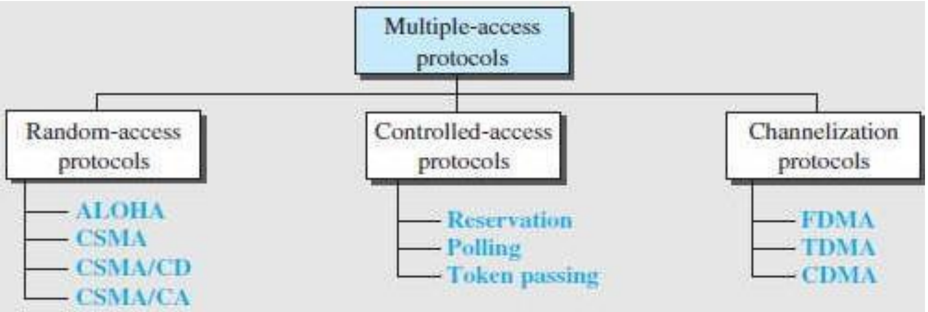


FIGURE: TAXONOMY OF MULTIPLE-ACCESS PROTOCOLS

❖ RANDOM ACCESS:

In **random-access** or **contention** methods, no station is superior to another station and none is assigned control over another. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send.

This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including testing the state of the medium.

Two features give this method its name. First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*. Second, no rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

In a random-access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict—***collision***— and the frames will be either destroyed or modified.

ALOHA:

Pure ALOHA:

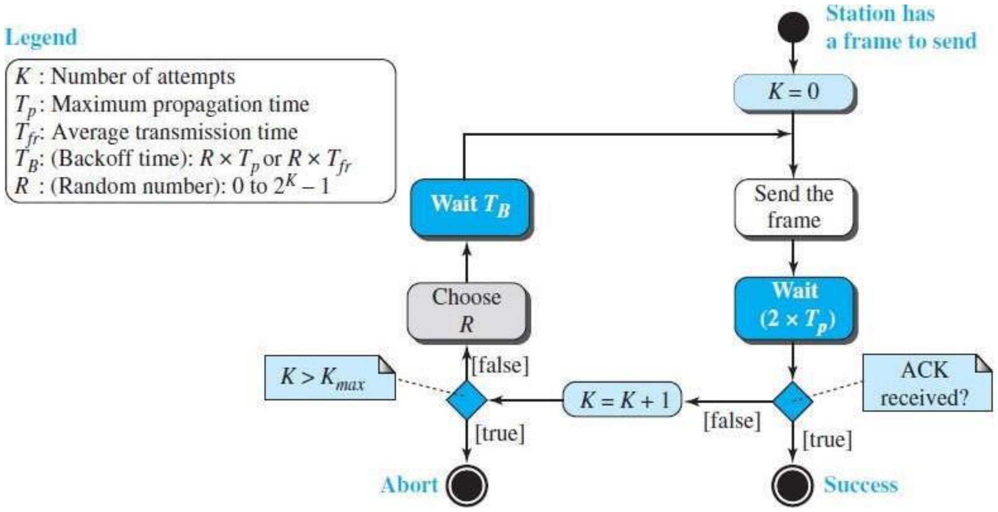
The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations.

The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the *backoff time* T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames.

After a maximum number of retransmission attempts K_{max}, a station must give up and try later.

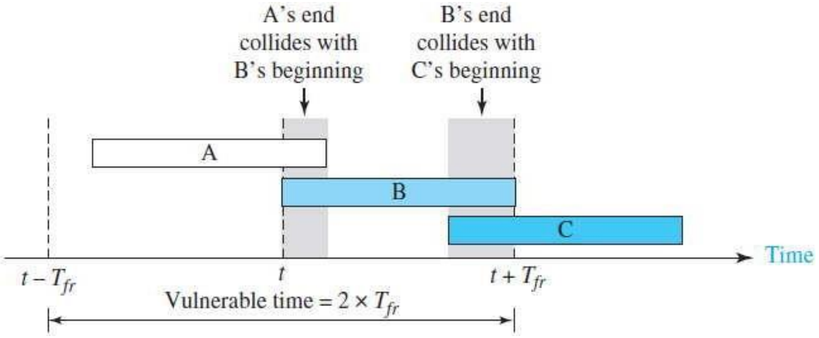


Legend
K : Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time
T_B: (Backoff time): $R \times T_p$ or $R \times T_{fr}$
R : (Random number): 0 to $2^K - 1$
Station has a frame to send
$K = 0$
Send the frame
Wait $(2 \times T_p)$
ACK received?
[true]
Success
[false]
$K = K + 1$
$K > K_{max}$
[false]
Choose R
Wait T_B
[true]
Abort

Fig: Procedure for pure ALOHA protocol

Vulnerable time

Let us find the ***vulnerable time,*** the length of time in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} seconds to send.



A's end collides with B's beginning
B's end collides with C's beginning
A
B
C
Time
$t - T_{fr}$
t
$t + T_{fr}$
Vulnerable time = $2 \times T_{fr}$

Station B starts to send a frame at time t. Now imagine station A has started to send its frame after $t - Tfr$. This leads to a collision between the frames from station B and station A. On the other hand, suppose that station C starts to send a frame before time $t + Tfr$. Here, there is also a collision between frames from station B and station C. Looking at Figure , we see that the vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

Pure ALOHA vulnerable time = 2 * *Tfr*

Throughput

Let us call G the average number of frames generated by the system during one frame transmission time. Then it can be proven that the average number of successfully transmitted frames for pure ALOHA is $S = G \times e-2G$. The maximum throughput $Smax$ is 0.184, for $G = 1/2$. In other words, if one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully. We expect $G = 1/2$ to produce the maximum throughput because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

Slotted ALOHA

Pure ALOHA has a vulnerable time of 2 × Tfr. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or just before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA. In **slotted ALOHA** we divide the time into slots of Tfr seconds and force the station to send only at the beginning of the time slot. The following Figure shows an example of frame collisions in slotted ALOHA.

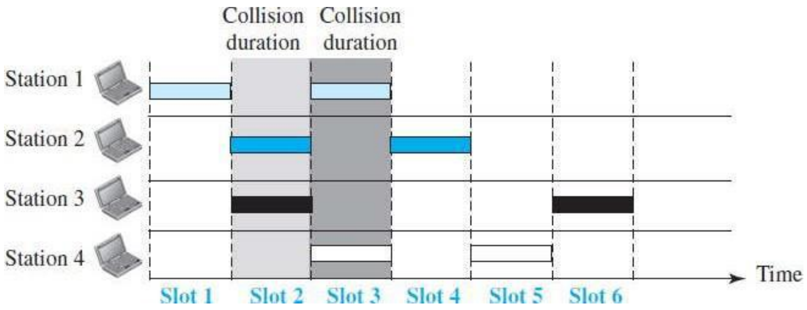


Fig:Frames in a slotted ALOHA network

Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has a lready finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one -half, equal to *Tfr*. The following Figure shows the situation.

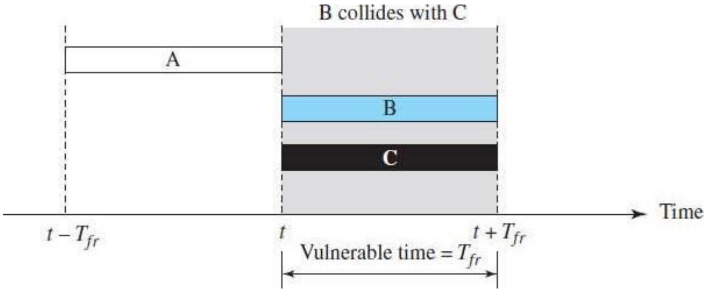


Fig:Vulnerable time for slotted ALOHA protocol

Throughput
It can be proven that the average number of successful transmissions for slotted ALOHA is *S* = *G* × *e*−*G*. The maximum throughput *Smax* is 0.368, when *G* = 1. In other words, if one frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. We expect *G* = 1 to produce maximum throughput because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

CSMA:

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. **Carrier sense multiple access (CSMA)** requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle “sense before transmit” or “listen before talk.” CSMA can reduce the possibility of collision, but it cannot eliminate it.

Persistence Methods: What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the **1-persistent method,** the **nonpersistent method,** and the ***p*-persistent method**

1-Persistent: The *1-persistent method* is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately. We will see later that Ethernet uses this method.

Nonpersistent: In the *nonpersistent method,* a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

p-*Persistent:* The p-*persistent method* is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The *p*-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability p, the station sends its frame.

2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.

a. If the line is idle, it goes to step 1.

b. If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.

CSMA/CD:

The CSMA method does not specify the procedure following a collision. **Carrier sense multiple access with collision detection (CSMA/CD)** augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

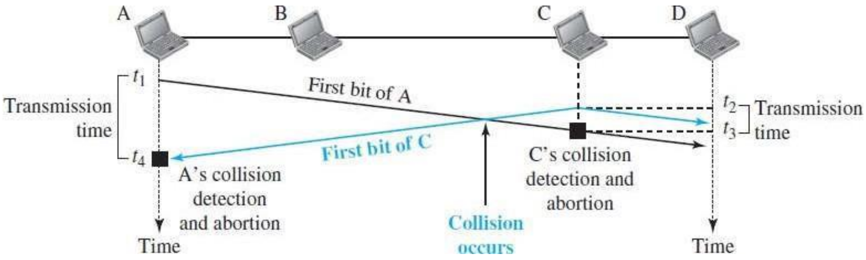


Fig: Collision of bits in CSMA/CD

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In the Figure , stations A and C are involved in the collision

At time $t1$, station A has executed its persistence procedure and starts sending the bits of its frame. At time $t2$, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time $t2$. Station C detects a collision at time $t3$ when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time $t4$ when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration $t4 - t1$; C transmits for the duration $t3 - t2$.

Minimum Frame Size: For CSMA/CD to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time *Tfr* must be at least two times the maximum propagation time *Tp*. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time *Tp* to reach the second, and the effect of the collision takes another time *TP* to reach the first. So the requirement is that the first station must still be transmitting after 2*Tp*.

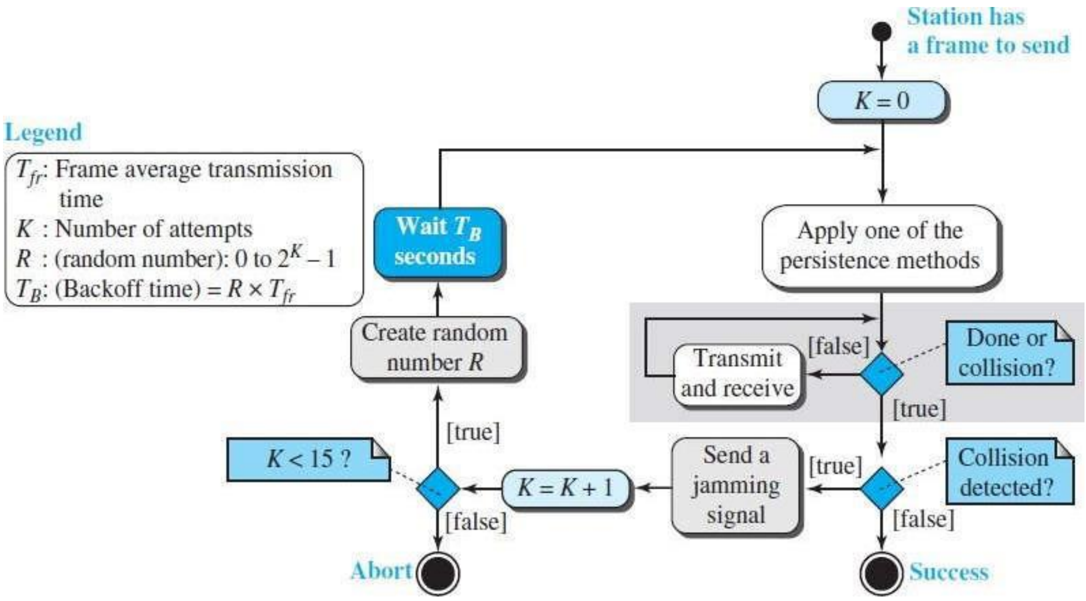
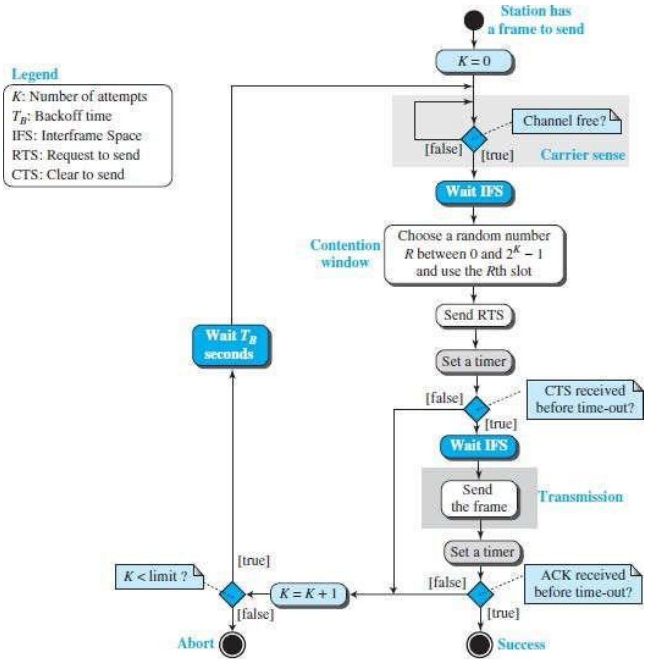


Fig: Procedure for CSMA/CD

<u>CSMA/CA</u>:

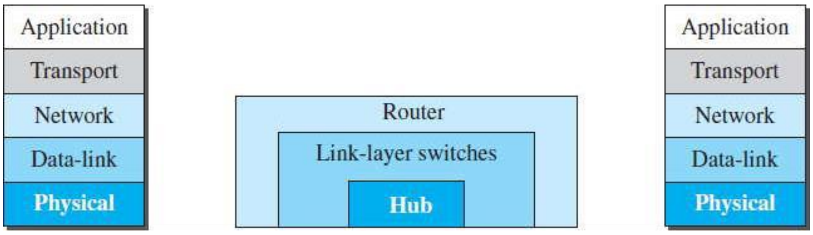
Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies: the interframe space, the contention window, and acknowledgments.

- ***Interframe Space (IFS).*** First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the ***interframe space*** or ***IFS***. Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station’s signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. After waiting an IFS time, if the channel is still idle, the station can send, but it still needs to wait a time equal to the contention window (described next). The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.
- ***Contention Window.*** The **contention window** is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential backoff strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the *p*-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.
- ***Acknowledgment.*** With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.



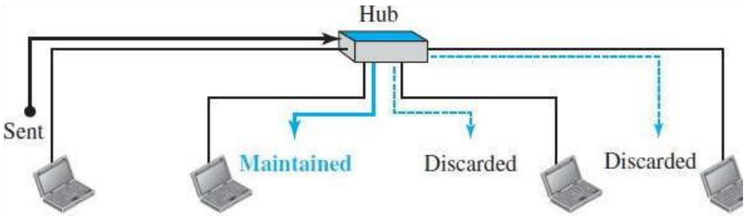
❖ CONNECTING DEVICES

connecting devices are used to connect hosts together to make a network or to connect networks together to make an internet. Connecting devices can operate in different layers of the Internet model. Three kinds of *connecting devices:* hubs, link-layer switches, and routers. Hubs today operate in the first layer of the Internet model. Link-layer switches operate in the first two layers. Routers operate in the first three layers.



Hubs

A **hub** is a device that operates only in the physical layer. A **repeater** receives a signal and, before it becomes too weak or corrupted, *regenerates* and *retimes* the original bit pattern. The repeater then sends the refreshed signal. In a star topology, a repeater is a multiport device, often called a *hub*, that can be used to serve as the connecting point and at the same time function as a repeater. Figure shows that when a packet sends from station A to station B arrives at the hub, the signal representing the frame is regenerated to remove any possible corrupting noise, but the hub forwards the packet from all outgoing ports except the one from which the signal was received. In other words, the frame is broadcast. All stations in the LAN receive the frame, but only station B keeps it. The rest of the stations discard it. Figure shows the role of a repeater or a hub in a switched LAN. The figure definitely shows that a hub does not have a filtering capability. It does not have the intelligence to find from which port the frame should be sent out. A hub or a repeater is a physical-layer device. They do not have a link-layer address and they do not check the link-layer address of the received frame. They just regenerate the corrupted bits and send them out from every port.

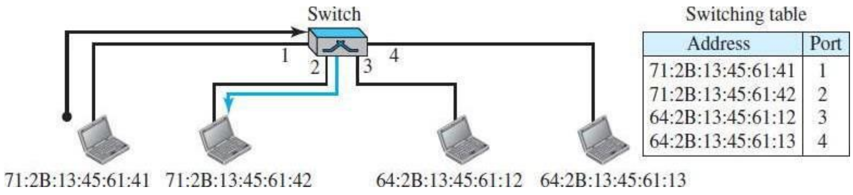


Link-Layer Switches

A **link-layer switch** (or ***switch***) operates in both the physical and the data-link layers. As a link-layer device, the link-layer switch can check the MAC addresses (source and destination) contained in the frame.

Filtering

The difference in functionality is between a link-layer switch and a hub is a link-layer switch has **filtering** capability. It can check the destination address of a frame and can decide from which outgoing port the frame should be sent. For example in Figure, we have a LAN with four stations that are connected to a link-layer switch. If a frame destined for station 71:2B:13:45:61:42 arrives at port 1, the link-layer switch consults its table to find the departing port. According to its table, frames for 71:2B:13:45:61:42 should be sent out only through port 2; therefore, there is no need for forwarding the frame through other ports.



Switching table

Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

Transparent Switches

A **transparent switch** is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet three criteria:

- Frames must be forwarded from one station to another.
- The forwarding table is automatically made by learning frame movements in the network.
- Loops in the system must be prevented.

Forwarding

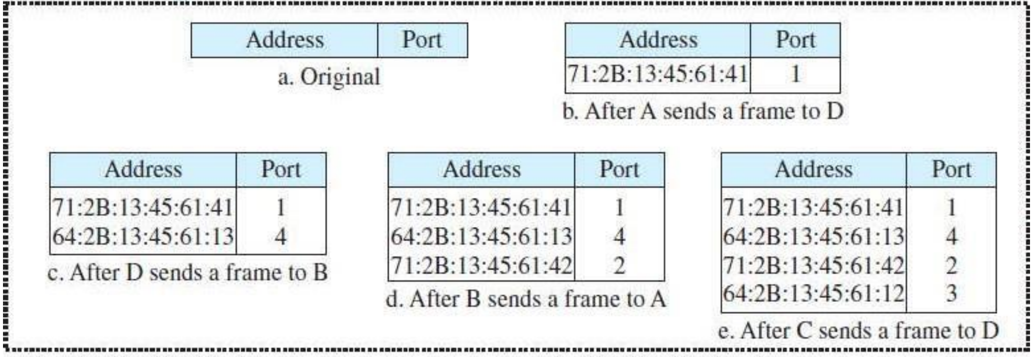
A transparent switch must correctly forward the frames, as discussed in the previous section.

Learning

The earliest switches had switching tables that were static. The system administrator would manually enter each table entry during switch setup. Although the process was simple, it was not practical. If a station was added or deleted, the table had to be modified manually. The same was true if a station's MAC address changed, which is not a rare event. For example, putting in a new network card means a new MAC address. A better solution to the static table is a dynamic table that maps addresses to ports (interfaces) automatically. To make a table dynamic, we need a switch that gradually learns from the frames' movements. To do this, the switch inspects both the destination and the source addresses in each frame that passes through the switch. The destination address is used for the forwarding decision (table lookup); the source address is used for adding entries to the table and for updating purposes. Let us elaborate on this process using Figure.

1. When station A sends a frame to station D, the switch does not have an entry for either D or A. The frame goes out from all three ports; the frame floods the netwo rk. However, by looking at the source address, the switch learns that station A must be connected to port 1. This means that frames destined for A, in the future, must be sent out through port 1. The switch adds this entry to its table. The table has its first entry now.
2. When station D sends a frame to station B, the switch has no entry for B, so it floods the network again. However, it adds one more entry to the table related to station D.
3. The learning process continues until the table has information about every port. However, note that the learning process may take a long time. For example, if a station does not send out a frame (a rare situation), the station will never have an entry in the table.

Gradual building of table



Address	Port

a. Original

Address	Port
71:2B:13:45:61:41	1

b. After A sends a frame to D

Address	Port
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4

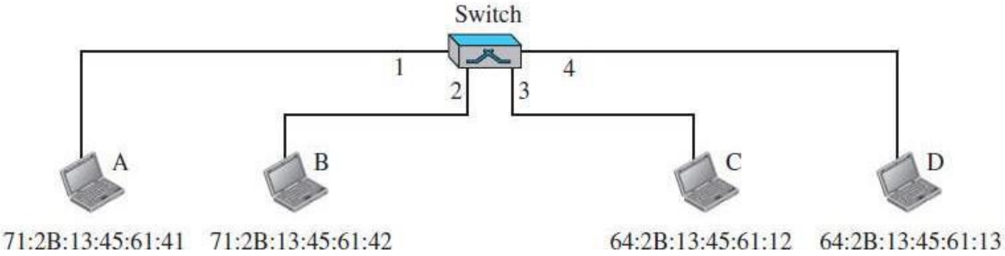
c. After D sends a frame to B

Address	Port
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4
71:2B:13:45:61:42	2

d. After B sends a frame to A

Address	Port
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3

e. After C sends a frame to D



Loop Problem:

Transparent switches work fine as long as there are no redundant switches in the system. Systems administrators, however, like to have redundant switches (more than one switch between a pair of LANs) to make the system more reliable. If a switch fails, another switch takes over until the failed one is repaired or replaced. Redundancy can create loops in the system, which is very undesirable. Loops can be created only when two or more broadcasting LANs (those using hubs, for example) are connected by more than one switch.

Figure shows a very simple example of a loop created in a system with two LANs connected by two switches.

1. Station A sends a frame to station D. The tables of both switches are empty. Both forward the frame and update their tables based on the source address A.

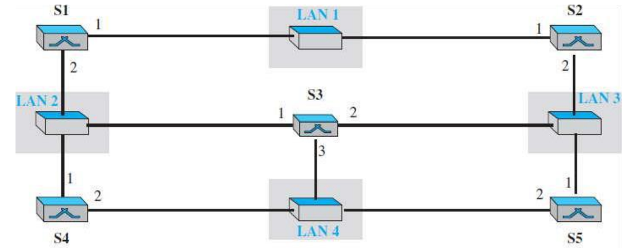
2. Now there are two copies of the frame on LAN 2. The copy sent out by the left switch is received by the right switch, which does not have any information about the destination address D; it forwards the frame. The copy sent out by the right switch is recei ved by the left switch and is sent out for lack of information about D.

3. Now there are two copies of the frame on LAN 1. Step 2 is repeated, and both copies are sent to LAN2.

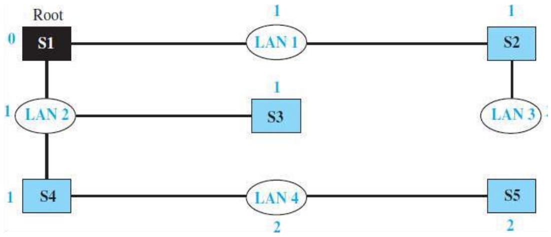
4. The process continues on and on. Note that switches are also repeaters and regener ate frames. So in each iteration, there are newly generated fresh copies of the frames.

Spanning Tree Algorithm

To solve the looping problem, the IEEE specification requires that switches use the spanning tree algorithm to create a loopless topology. In graph theory, a **spanning tree** is a graph in which there is no loop.



a. Actual system



Routers

A **router** is a three-layer device; it operates in the physical, data-link, and network layers. As a physical- layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network - layer device, a router checks the network-layer addresses. A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet. There are three major differences between a router and a repeater or a switch.

1. A router has a physical and logical (IP) address for each of its interfaces.

2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.

3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Unit-III The Network Layer

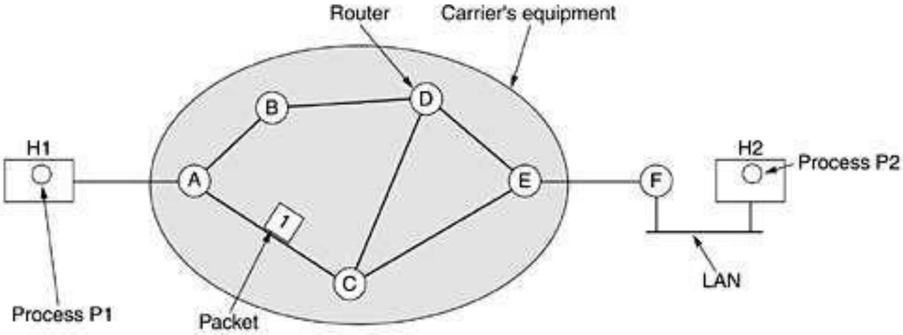
❖ Network Layer Design Issues:

While designing the network layer we have to consider some of the design issues. These issues include the service provided to the transport layer and the internal design of the subnet.

Store-and-Forward Packet Switching:

The major components of the system are the carrier's equipment, shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host *H1* is directly connected to one of the carrier's routers, *A*, by a leased line. In contrast, *H2* is on a LAN with a router, *F*, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown *F* as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers.

Figure 5-1. The environment of the network layer protocols.



Here, a host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is ***stored*** there until it has fully arrived so the checksum can be verified. Then it is ***forwarded*** to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is ***store-and-forward*** packet switching.

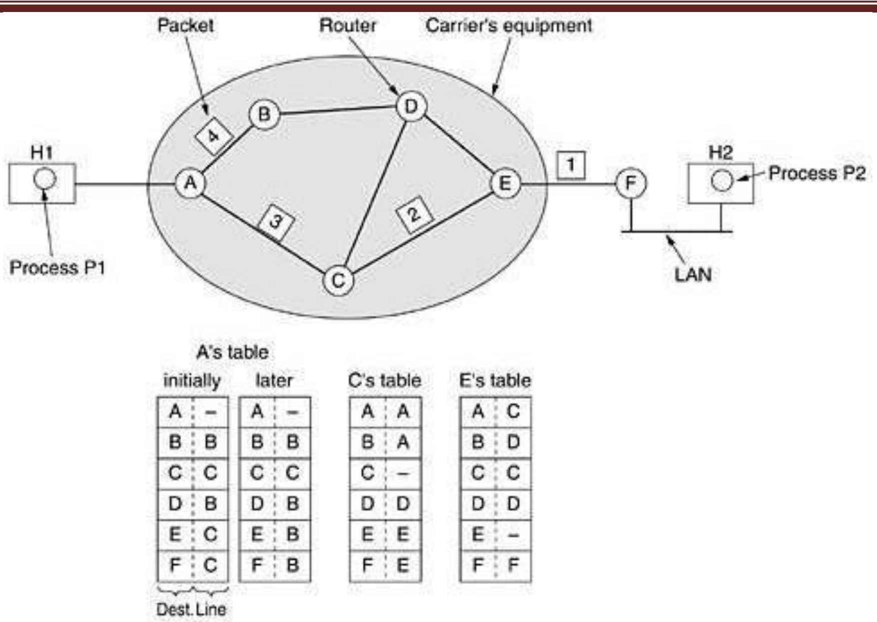
Services Provided to the Transport Layer:

The network layer provides services to the transport layer at the ***network layer/transport layer interface***. The network layer services have been designed with the following goals in mind.

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

Implementation of Connectionless Service:

Figure : Routing within a datagram subnet.



A's table initially (Dest.)	(Line)	A's table later (Dest.)	(Line)	C's table (Dest.)	(Line)	E's table (Dest.)	(Line)
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

Dest. Line

In connectionless service, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** and the subnet is called a **datagram subnet**.

Every router has an internal table telling it *where to send packets* for each possible destination. Each table entry is a pair consisting of a ***destination*** and ***the outgoing line*** to use for that destination. Only directly-connected lines can be used.

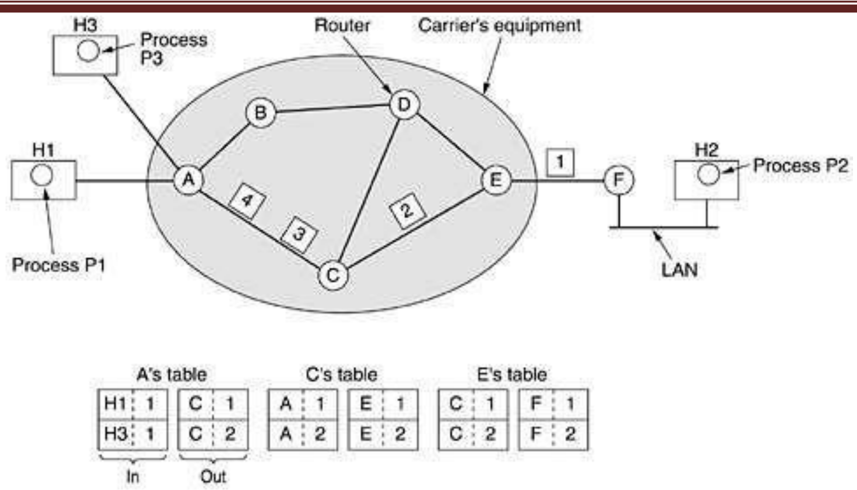
As they arrived at *A*, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to *C* according to *A*'s table. Packet 1 was then forwarded to *E* and then to *F*. When it got to *F*, it was encapsulated in a data link layer frame and sent to *H2* over the LAN. Packets 2 and 3 follow the same route However, something different happened to packet 4. When it got to *A* it was sent to router *B*, even though it is also destined for *F*. The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**.

Implementation of Connection-Oriented Service:

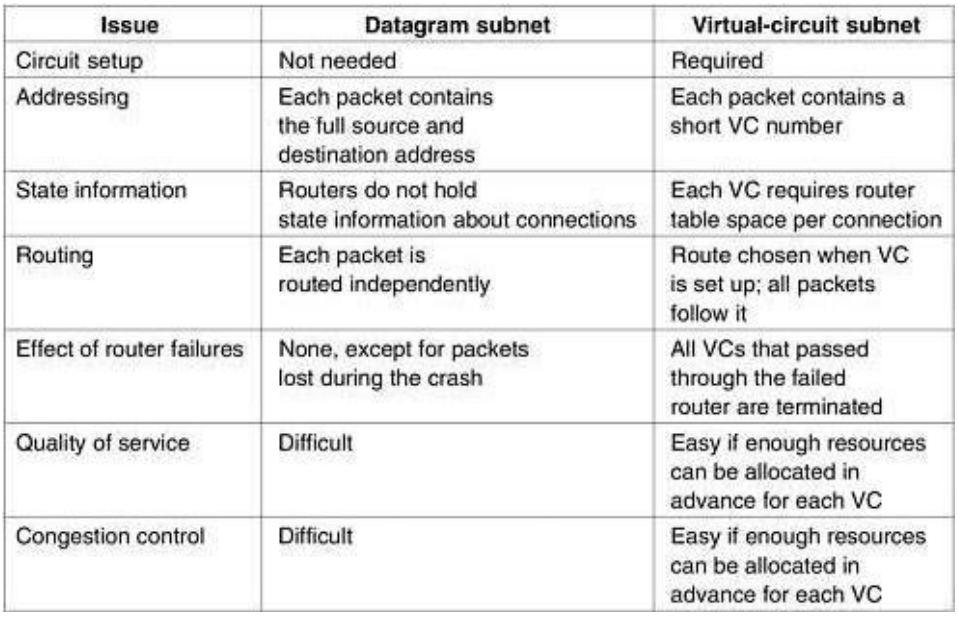
If *connection-oriented service* is used, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), in analogy with the physical circuits set up by the telephone system, and the subnet is called a **virtual-circuit subnet**.

As an example, consider the situation of Fig. 5-3. Here, host *H1* has established connection 1 with host *H2*. It is remembered as the first entry in each of the routing tables. The first line of *A*'s table says that if a packet bearing connection identifier 1 comes in from *H1*, it is to be sent to router *C* and given connection identifier 1. Similarly, the first entry at *C* routes the packet to *E*, also with connection identifier 1.

Figure: Routing within a virtual-circuit subnet.



Comparison of Virtual-Circuit and Datagram Subnets:



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

❖ **Routing Algorithms:**

Characteristics of routing algorithms:

- **Correctness:** it could able to deliver packets from source to destination without failure or without other nodes.
- **Simplicity:** the function should be simple in operation.
- **Robustness:** if the network is delivering packets via some route, if any failures or overloads occur, the function should react to such contingencies without the loss of packets or the breaking of virtual circuits.
- **Stability:** The outing function should react to contingencies slowly that are neither fast nor too slow. Why means, for example, if the network may react to congestion in one area by shifting most of load to second area. Now the second area is overloaded and the first is under-utilized, causing a second shift. During these shifts, packets may travel in loops through the network.
- **Fairness and Optimality:** some performance criteria may give higher priority to the exchange of packets between neighbor stations compared to an exchange between distant stations. This policy may maximize average throughput but will appear unfair to the station that primarily needs to communicate with distant stations.

- **Efficiency:** The efficiency routing function involves the processing overhead at each node and often a transmission overhead.

Classification of routing algorithms:

Routing algorithms can be grouped into two major classes:

- **Nonadaptive algorithms** do not base their *routing decisions* on measurements or estimates of the *current traffic* and *topology*. Instead, the choice of the route to use to get from *I* to *J* (for all *I* and *J*) is computed in advance, off-line, and downloaded to the routers when the network is booted. This procedure is sometimes called ***"static routing"***.

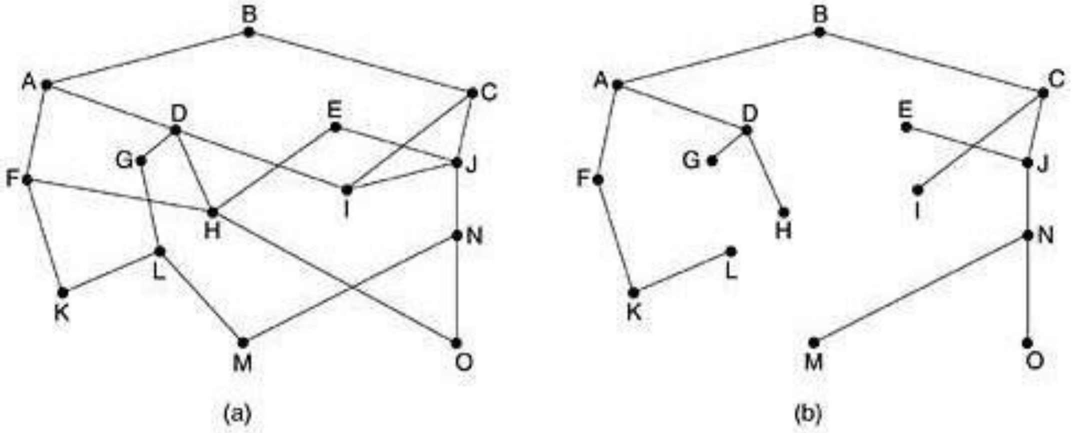
- **Adaptive algorithms** on the contrary are ***dynamic*** and ***online***. They collect their information about the state of the network and make routing decisions based on the latest information, for example, Distance vector routing and link state routing.

The Optimality Principle:

It states that one can make a general statement about *optimal routes* without regard to network topology or traffic. This statement is known as the **optimality principle**. 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* r_1 and the rest of the route r_2. If a route better than r_2 existed from *J* to *K*, it could be concatenated with r_1 to improve the route from *I* to *K*, *contradicting* our statement that $r1r2$ is optimal.

As the result of optimality principle 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. 5-6, where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers. 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.

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

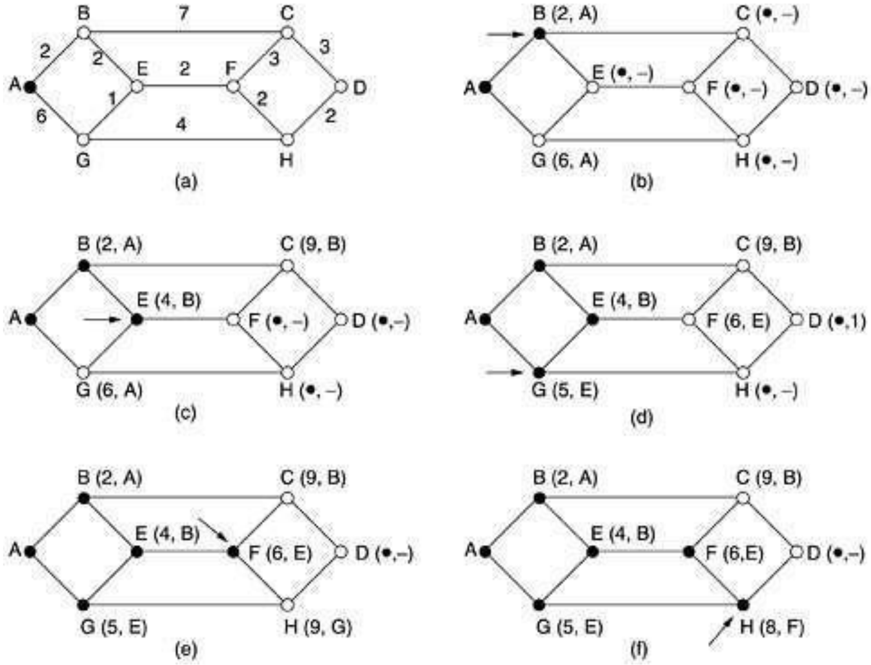


Shortest Path Routing:

The idea behind routing algorithms is to build a ***graph*** of the subnet, with each *node* of the graph representing a *router* and each *arc* of the graph representing a *communication line*. To choose a route between a given pair of routers, the *algorithm just finds the shortest path* between them on the graph.

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the *number of hops*. Using this metric, the paths *ABC* and *ABE* in Fig. 5-7 are equally long. Another metric is the *geographic distance in kilometers*, in which case *ABC* is clearly much longer than *ABE*. Another metric is may be *time delay* etc., like so many metrics can be used for shortest path routing.

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



Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to ***Dijkstra*** (1959). Each *node is labeled* (in parentheses) with its *distance from the source node along the best known path*. Initially, no paths are known, so all nodes are labeled with *infinity*. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either ***tentative*** *or* ***permanent***.

⇨ Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

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

We now start at *B* and examine all nodes adjacent to it. If the sum of the label on *B* and the distance from *B* to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.

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

Flooding:

Flooding is a static algorithm, in which 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 (Discourage) the process.

- ✓ One such measure is to have a *hop counter* contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.
  - o 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. 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.

Applications and Advantages:

- ⇨ Flooding is very effective routing approach, when, the information in the routing tables is not available, such as during ***system start up***.
- ⇨ Flooding is also effective when the source needs to send a packet to all hosts connected to the network for example in ***military applications***.
- ⇨ In ***distributed data base applications***, it is sometimes necessary to update the entire database concurrently; in such cases flooding is used.
- ⇨ Flooding always chooses the ***shortest path***, because it chooses every possible path in parallel.
- ⇨ In ***wireless networks***, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property.

Distance Vector Routing:

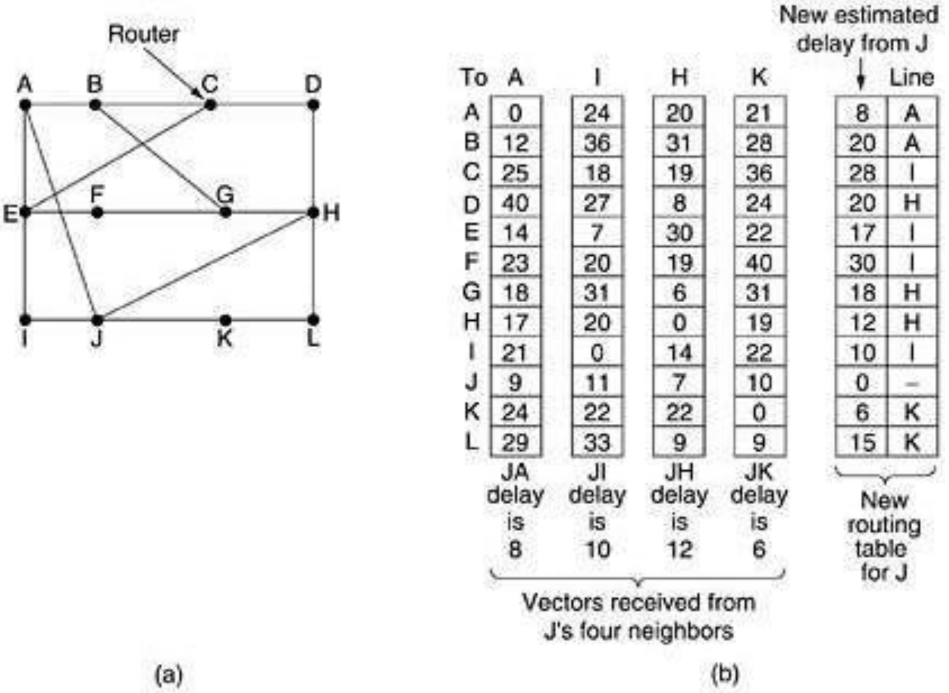
Modern computer networks generally use *dynamic routing algorithms* rather than the *static* ones because static algorithms *do not take the current network load into account*.

Distance vector routing algorithms operate by having each router maintain a ***table*** (i.e, a vector) giving the best known distance to each destination and which line to use to get there. These tables are updated by *exchanging information with the neighbors*.

The distance vector routing algorithm is also called by other names as, distributed **Bellman-Ford routing algorithm** and the **Ford-Fulkerson algorithm**.

This can be explained in Fig. Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbors of router *J*. 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.

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



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

Vectors received from J's four neighbors

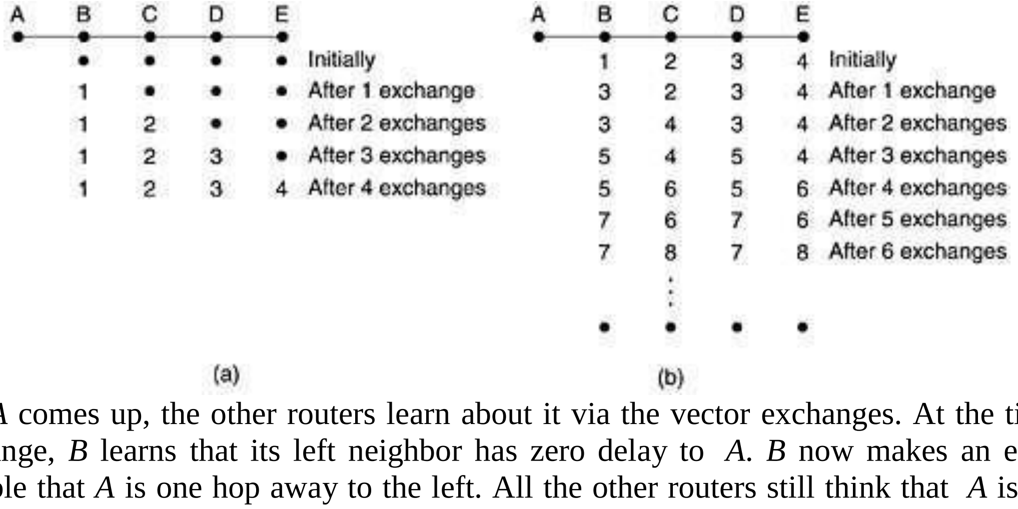
(b)

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 can 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 Count-to-Infinity Problem:

consider the five-node (linear) subnet of Fig (a), where the delay metric is the number of <u>*hops*</u>. 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.

Figure: The count-to-infinity problem.



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

(a)

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

(b)

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* now 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. 5-10(a). Clearly, the good news is spreading at the rate of one hop per exchange, then, C,D,E routers are updated as 2,3,4 respectively.

Now let us consider the situation of Fig. (b), in which all the lines and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4, respectively. Suddenly *A* goes down, or alternatively, the line between *A* and *B* is cut, which is effectively the same thing from *B*'s point of view.

- When **A** goes down, line between **A** and **B** is out.
- <u>**B** does not hear anything from **A**</u>**.**
- **C** informs **B**, I Have Path to **A**, of length 2.
- If metric used is Time Delay, there is no well-defined upper bound. So, high value is needed to prevent a path with a long delay from being treated as down. This problem is known as **count-to-infinity**.

<u>Link State Routing:</u>

Distance vector routing was replaced by ***link state routing***. Two primary problems caused its demise.

- <u>First</u>, since the delay metric was *<u>queue length</u>*, it did not take line bandwidth into account when choosing routes.
- Second problem is the count-to-infinity problem.

The idea behind link state routing is simple and can be stated as five parts. Each router must 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.

Learning about the Neighbors:

When a router is booted, its first task is to *<u>learn</u>* who its neighbors are. It accomplishes this goal by sending a special <u>**HELLO**</u> packet on each point-to-point line.

Measuring Line Cost:

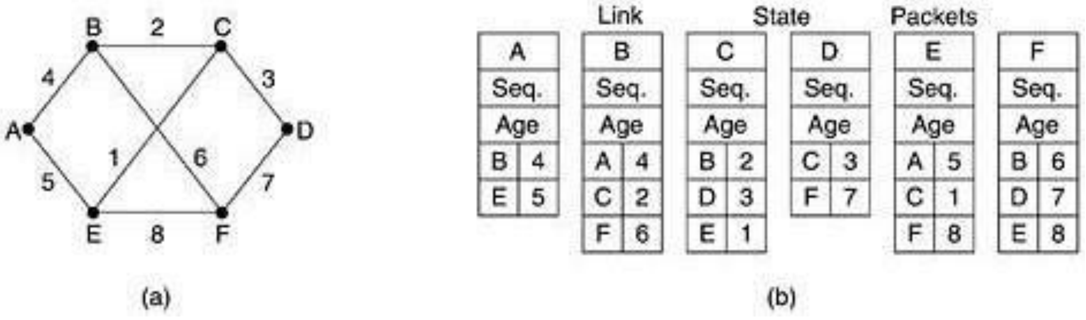
The <u>link state routing algorithm</u> requires each router to know, or at least have a reasonable *<u>estimate</u>* of, the *<u>delay</u>* to each of its neighbors. The most direct way to determine this delay is to send over the line a ***<u>special ECHO packet</u>*** that the other side is required to send back immediately. By measuring the *round-trip time* and *dividing it by two*, the sending router can get a reasonable estimate of the delay.

Building Link State Packets:

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

An example subnet is given in Fig. (a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig.(b).

Figure. (a) A subnet. (b) The link state packets for this subnet.



Link State Packets

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

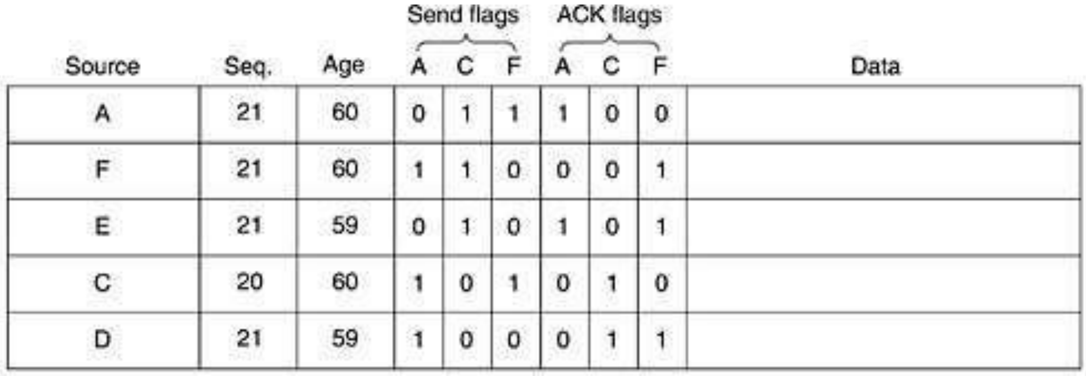
Distributing the Link State Packets:

The next step after building link state packets is to distribute them across the network. Flooding is used as the basic algorithm for distributing link state packets. To avoid flooding the same packet, each new packet is given a sequence number. When a packet arrives at a router for flooding then it checks whether this packet is already seen by using a pair (*source router*, *sequence number*) that each router have.

When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded. If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete (**outdated**) since the router has more recent data. The ***age*** of each packet *decrement* it once per second. When the *age hits zero*, the information from that router is discarded.

The ***data structure*** used by router 'B' for the subnet is depicted in Fig. Each **row** here corresponds to a recently-arrived, but as yet not fully-processed, link state packet. The table records where the *packet originated*, its *sequence number* and *age*, and the data. In addition, there are send and acknowledgement flags for each of B's three lines (to A, C, and F, respectively). The ***send flags*** mean that the packet must be sent on the indicated line. The ***acknowledgement flags*** mean that it must be acknowledged there.

Figure :The packet buffer for router B



			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

In above table, the link state packet from *A* arrives directly, so it must be sent to *C* and *F* and acknowledged to *A*, as indicated by the flag bits. Similarly, the packet from *F* has to be forwarded to *A* and *C* and acknowledged to *F*.

***Computing the New Routes*:**

Once a router has accumulated a full set of link state packets, router builds 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.

Now ***Dijkstra's algorithm*** can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed (continued).

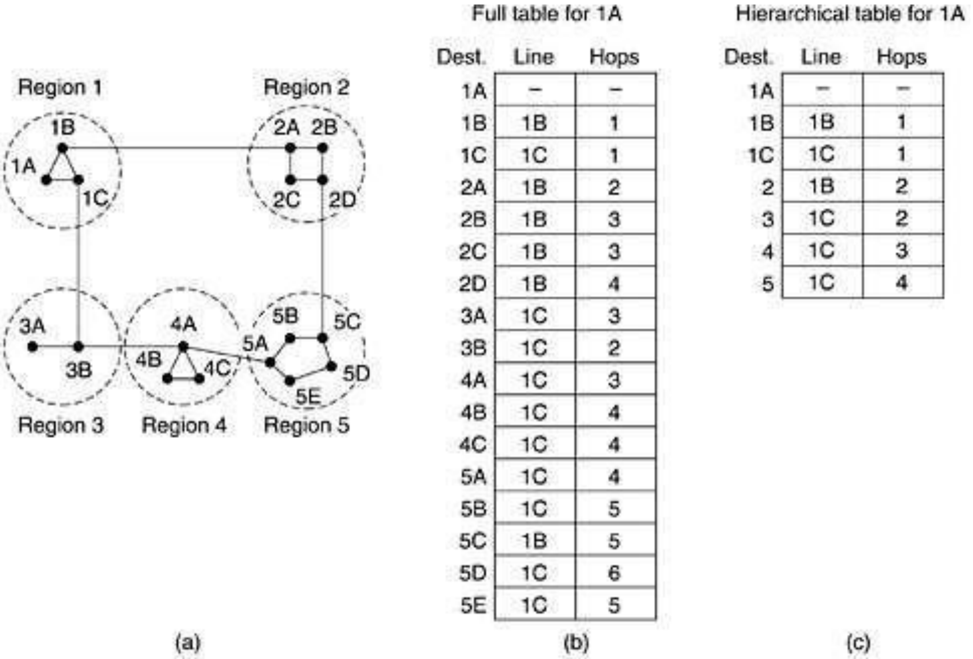
Hierarchical Routing:

Hierarchical routing is an algorithm for routing packets hierarchically. It is used due to the following reasons.

- As networks grow in size, the router routing tables grow proportionall y.
- Router memory consumed by ever-increasing tables.
- More CPU time is needed to scan them and more bandwidth is needed to send status reports about them.
- At a certain point the network may grow to the point where it is no longer feasible for every router to have an entry for every other router.

When hierarchical routing is used, the routers are divided into **regions**, with each router knowing all the details about *how to route packets to destinations within its own region*, but knowing *nothing* *about the internal structure of other regions*. When different networks are interconnected, then each one can be treated as a separate region in order to free the routers in one network from having to know the topological structure of the other ones.

Figure : Hierarchical routing.



Full table for 1A

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

Hierarchical table for 1A

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

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

<u>Broadcast Routing:</u>

Sending a packet to all destinations simultaneously is called ***broadcasting***; the algorithms used for broadcasting are called ***broadcast routing***. Various methods have been proposed for doing it. They are,

1) Distinct point-to-point routing
2) Flooding
3) Multi-destination routing
4) Use of spanning tree
5) Reverse path forwarding

Distinct point-to-point routing:

This is the simplest method for broadcasting.in this method '*sender simply sends a distinct packet to each destination or to all the nodes in the network'*. Thus it takes no special features of the subnet. This method is not desirable due to two reasons. First, Wasteful of bandwidth, Second it requires the source to have a complete list of all destinations.

Flooding:

Flooding is another obvious candidate. This algorithm sends a packet on every outgoing line except the line on which it arrived. The problem with flooding as a broadcast technique is the same problem it has as a point-to-point routing algorithm: "it generates too many packets and consumes too much bandwidth".

Multi-destination routing:

A third algorithm is **multi-destination routing**. If this method is used, each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed. The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line. In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.

Use of spanning tree:

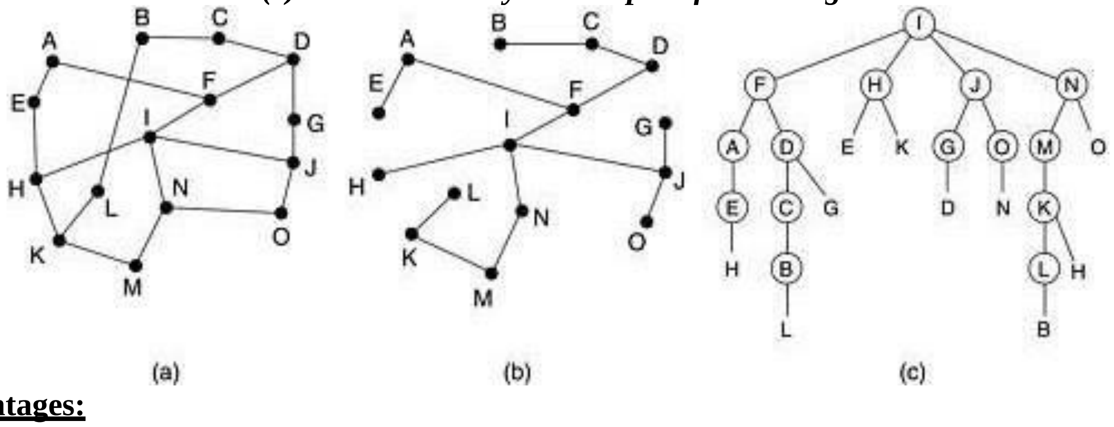
A **spanning tree** is a subset of the subnet that includes all the routers but contains ***no loops***.

If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on. This method makes <u>*excellent use of bandwidth*</u>, generating the <u>*absolute minimum number of packets*</u> necessary to do the job.

Reverse path forwarding:

An example of reverse path forwarding is shown in Fig. Part (a) shows a *subnet*, part (b) shows a *sink tree for router I* of that subnet, and part (c) shows how the reverse path algorithm works. On the first hop, *I* send packets to *F*, *H*, *J*, and *N*, as indicated by the second row of the tree. Each of these packets arrives on the preferred path to *I* (assuming that the preferred path falls along the sink tree) and is so indicated by a *circle* around the letter. On the second hop, eight packets are generated, two by each of the routers that received a packet on the first hop. As it turns out, all eight of these arrive at previously unvisited routers, and five of these arrive along the preferred line. Of the six packets generated on the third hop, only three arrive on the preferred path (at *C*, *E*, and *K*); the others are duplicates. After five hops and 24 packets, the broadcasting terminates, compared with four hops and 14 packets had the sink tree been followed exactly.

Figure. Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding.



<u>**Advantages:**</u>

- ➢ The reverse path forwarding is that it is both reasonably efficient and easy to imple ment.
- ➢ It does not require routers to know about spanning trees.
- ➢ It does not have the overhead of a destination list or bit map in each broadcast packet as does multi-destination addressing.
- ➢ It does not require any special mechanism to stop the process.

<u>**Multicast Routing:**</u>

For some applications such as tele conferencing, a source may want to send packets to multiple destinations simultaneously or a group of processes implementing a distributed database systems. It is frequently necessary for one process to send a message to all the other members of the group.

⇨ If the group is small, it can just send each other member a point-to-point message.

⇨ If the group is large, this strategy is expensive.

Thus, we need a way to send messages to well defined groups that are numerically large in size but small compared to the network as a whole.

Sending a message to such a group is called **multicasting**, and its routing algorithm is called **multicast routing**.

Multicasting requires group management.

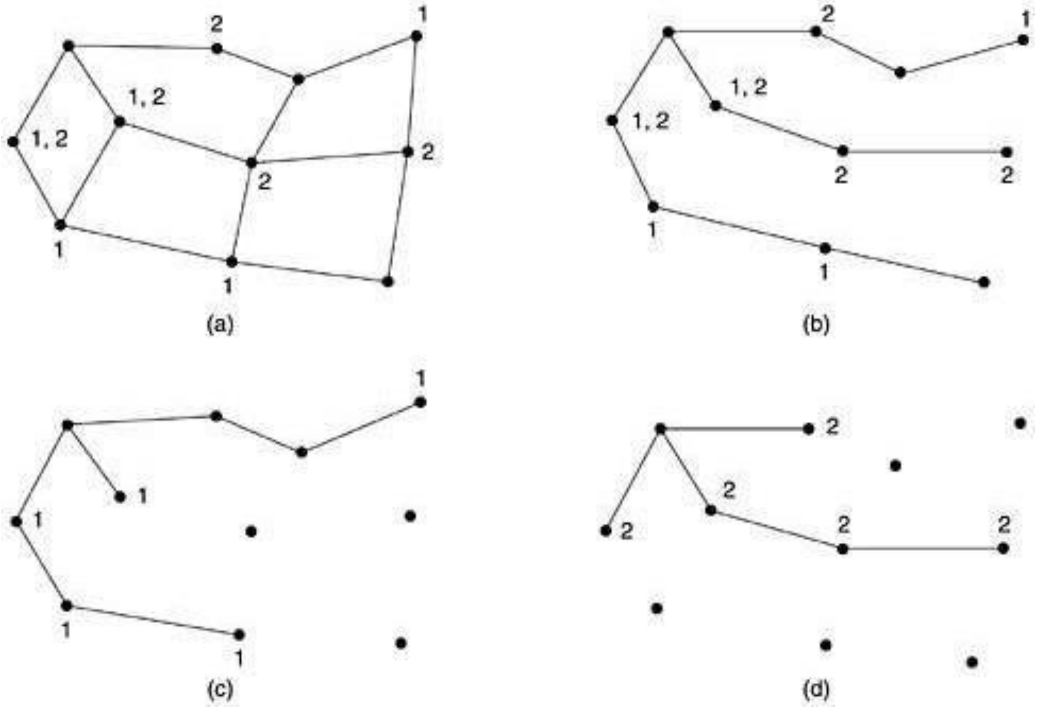
✓ To *create* and *destroy groups* and

✓ To allow processes to *join* and *leave* groups.

The routing algorithm does not know how these tasks are accomplished but when a process joins a group; it informs its host of this fact. It is important that routers know which of their hosts belong to which groups. Either host must inform their routers about changes in group membership, or routers must query their hosts periodically. Either way, routers learn about which of their hosts are in which groups. Routers tell their neighbors, so the information pr opagates through the subnet.

To do multicast routing, each router computes a *spanning tree* covering all other routers. For example, in Fig.(a) we have two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig. (b).

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

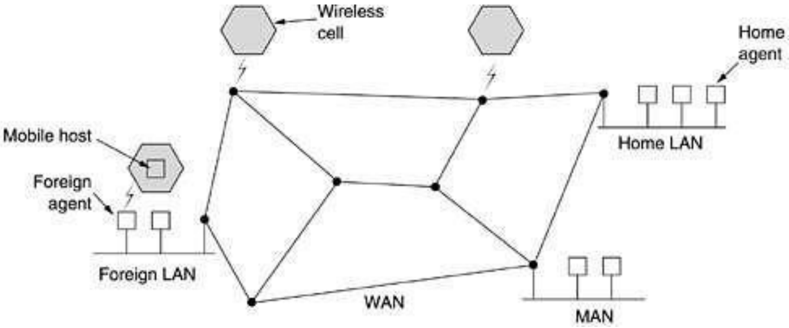


When a process sends a multicast packet to a group, the **first** router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are members of the group. In our example, Fig.(c) shows the pruned spanning tree for group 1. Similarly, Fig. (d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate s panning tree.

Various ways of pruning the spanning tree are possible. The simplest one can be used if *link state routing* is used and each router is aware of the complete topology, including which hosts belong to which groups. Then the spanning tree can be pruned, starting at the end of each path, working toward the root, and removing all routers that do not belong to the group in question.

<u>Routing for Mobile Hosts:</u>

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

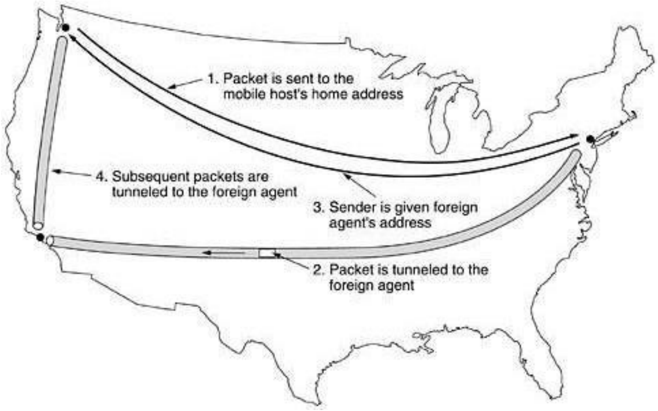


In the model of Fig, the world is divided up (geographically) into small units. Let us call them areas, where an area is typically a LAN or wireless cell. Each area has one or more **foreign agents**, which are processes that keep track of all mobile hosts visiting the area. In addition, each area has a **home agent**, which keeps track of hosts whose home is in the area, but who are currently visiting another area.

When a new host enters an area, either by connecting to it (e.g., plugging into the LAN) or just wandering (travelling) 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 for eign 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 to convince the home agent that the mobile host is really there.
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 now registered.

Figure: Packet routing for mobile hosts.


1. Packet is sent to the mobile host's home address
4. Subsequent packets are tunneled to the foreign agent
3. Sender is given foreign agent's address
2. Packet is tunneled to the foreign agent

Ideally, when a host leaves an area, that, too, should be announced to allow deregistration, but 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 because that is what the address says should be done, as illustrated in step 1 of Fig.

The home agent then does two things.

- **First**, it encapsulates the packet in the payload field of an outer packet and sends the latter to the foreign agent (step 2). This mechanism is called ***tunneling***;
- **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.

Routing in Ad Hoc Networks:

Here routing can be done when the hosts are mobile and also routers themselves are mobile. Among the possibilities are:

1. Military vehicles on a battlefield with no existing infrastructure.
2. A fleet of ships at sea.
3. Emergency workers at an earthquake that destroyed the infrastructure.
4. A gathering of people with notebook computers in an area lacking 802.11.

In all these cases, and others, each node consists of a *router* and a *host*, usually on the same computer. *Networks of nodes* that just happen to be near each other are called **ad hoc networks** or **MANETs** (**Mobile Ad hoc Networks**).

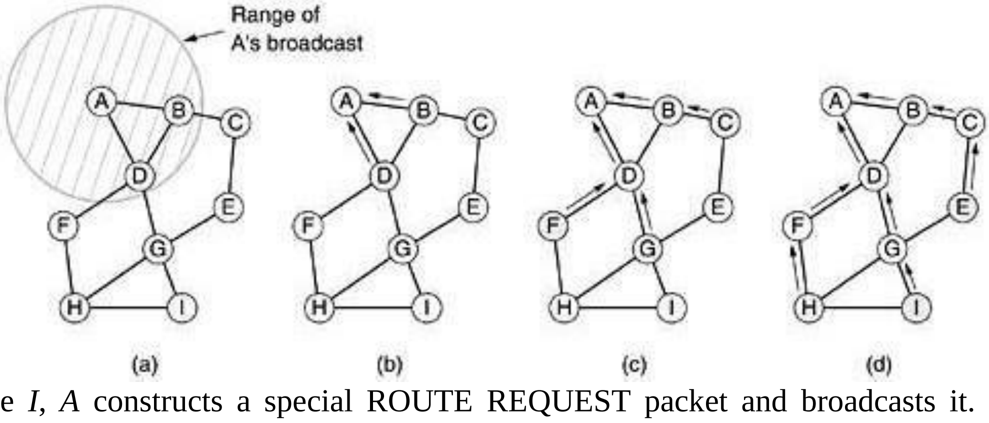
A variety of routing algorithms for ad hoc networks have been proposed. One of the more interesting ones is the ***AODV (Adhoc On-demand Distance Vector) routing algorithm***. It is a distant relative of the Bellman-Ford distance vector algorithm but adapted to work in a mobile environment and takes into account the *limited bandwidth* and *low battery life* found in this environment.

- It is an ***on-demand algorithm***, that is, it determines a route to some destination only when somebody wants to send a packet to that destination.

Route Discovery:

Consider the ad hoc network of Fig., in which a process at node *A* wants to send a packet to node ***I***. The AODV algorithm maintains a table at each node, keyed (enetered) by destination, giving information about that destination, including which neighbor to send packets to in order to reach the destination. Suppose that *A* looks in its table and does not find an entry for *I*. It now has to discover a route to *I*. This property of discovering routes only when they are needed is what makes this algorithm **"on demand."**

Figure: (a) Range of* A*'s broadcast. (b) After* B *and* D *have received* A*'s broadcast
(c) After* C, F, *and* G *have received* A*'s broadcast.
(d) After* E, H, *and* I *have received* A*'s broadcast.
The shaded nodes are new recipients. The arrows show the possible reverse routes.



To locate *I*, *A* constructs a special ROUTE REQUEST packet and broadcasts it. The packet reaches *B* and *D*, as illustrated in Fig. (a). In fact, the reason *B* and *D* are connected to *A* in the graph is that they can receive communication from *A*. *F*, for example, is not shown with an arc to *A* because it cannot receive *A*'s radio signal. Thus, *F* is not connected to *A*.

Route Maintenance:

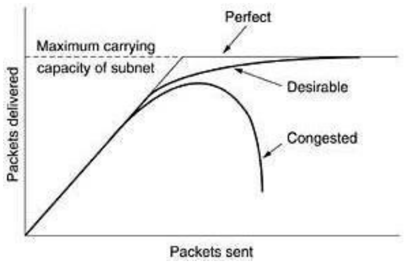
Because nodes can move or be switched off, the topology can change spontaneously. Periodically, each node broadcasts a ***Hello*** message. Each of its neighbors is expected to respond to it. If no response is forthcoming, the broadcaster knows that that neighbor has moved out of range and is no longer connected to it. Similarly, if it tries to send a packet to a neighbor that does not respond, it learns that the neighbor is no longer available.

This information is used to purge (**remove**) routes that no longer work. For each possible destination, each node, *N*, keeps track of its neighbors that have fed it a packet for that destination during the last Δ*T* seconds. These are called *N*'s **active neighbors** for that destination.

❖ Congestion Control Algorithms:

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

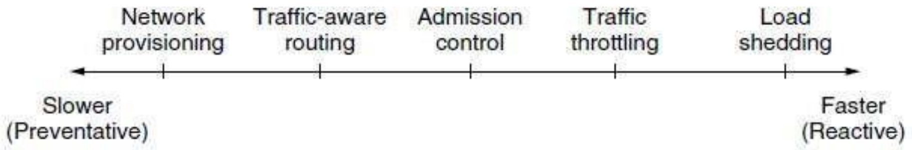
Figure depicts the onset of congestion. When the number of packets hosts send into the network is well within its carrying capacity, the number delivered is proportional to the number sent. If twice as many are sent, twice as many are delivered. However, as the offered load approaches the carrying capacity, bursts of traffic occasionally fill up the buffers inside routers and some packets are lost. These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is now congested.



If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up. If there is insufficient memory to hold all of them, packets will be lost. Adding more memory may help up to a point that if routers have an infinite amount of memory, congestion gets worse, not better. This is because by the time packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. This makes matters worse, not better—it leads to congestion collapse.

Low-bandwidth links or routers that process packets more slowly than the line rate can also become congested. In this case, the situation can be improved by directing some of the traffic away from the bottleneck to other parts of the network. Eventually, however, all regions of the network will be congested.

Approaches to Congestion Control



The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion, for example, turning on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. This is called **provisioning** and happens on a time scale of months, driven by long-term traffic trends.

Some local radio stations have helicopters flying around their cities to report on road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called **traffic-aware routing**. Splitting traffic across multiple paths is also helpful.

Sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called **admission control**.

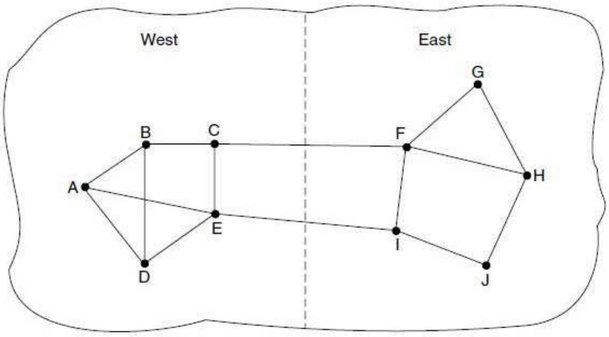
At a finer granularity, when congestion is imminent the network can deliver feedback to the sources whose traffic flows are responsible for the problem. The network can request these sources to throttle their traffic, or it can slow down the traffic itself. Two difficulties with this approach are how to identify the onset of congestion, and how to inform the source that needs to slow down.

To tackle the first issue, routers can monitor the average load, queueing delay, or packet loss. In all cases, rising numbers indicate growing congestion. To tackle the second issue, routers must participate in a feedback loop with the sources.

Finally, when all else fails, the network is forced to discard packets that it cannot deliver. The general name for this is **load shedding**. A good policy for choosing which packets to discard can help to prevent congestion collapse.

Traffic-Aware Routing

The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion. The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favour paths that are more lightly loaded, all

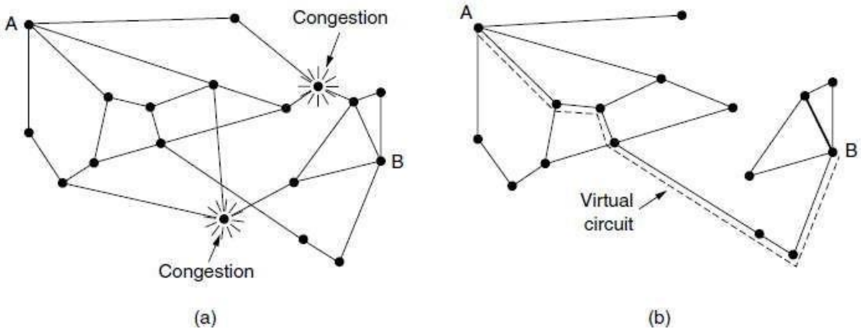


Consider the network of Fig. which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queuing delay in the weight used for the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, loading this link. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.

Admission Control

By analogy, in the telephone system, when a switch gets overloaded it practices admission control by not giving dial tones. The task is straightforward in the telephone network because of the fixed bandwidth of calls (64 kbps for uncompressed audio). However, virtual circuits in computer

networks come in all shapes and sizes. Thus, the circuit must come with some characterization of its traffic if we are to apply admission
control.



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

Traffic Throttling

When congestion is imminent, it must tell the senders to throttle back their transmissions and slow down. The term **congestion avoidance** is sometimes used to contrast this operating point with the one in which the network has become (overly) congested.

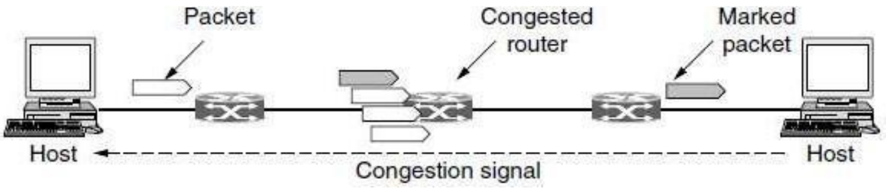
Choke Packets

The most direct way to notify a sender of congestion is to tell it directly. In this approach, the router selects a congested packet and sends a **choke packet** back to the source host, giving it the destination found in the packet. The original packet may be tagged (a header bit is turned on) so that it will not generate any
more choke packets farther along the path and then forwarded in the usual way. To avoid increasing load on the network during a time of congestion, the router may only send choke packets at a low rate.

When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination, for example, by 50%. In a datagram network, choke packets to be sent to fast senders, because they will have the most packets in the queue. The host should ignore these additional chokes for the fixed time interval until its reduction in traffic takes effect. After that period, further choke packets indicate
that the network is still congested.

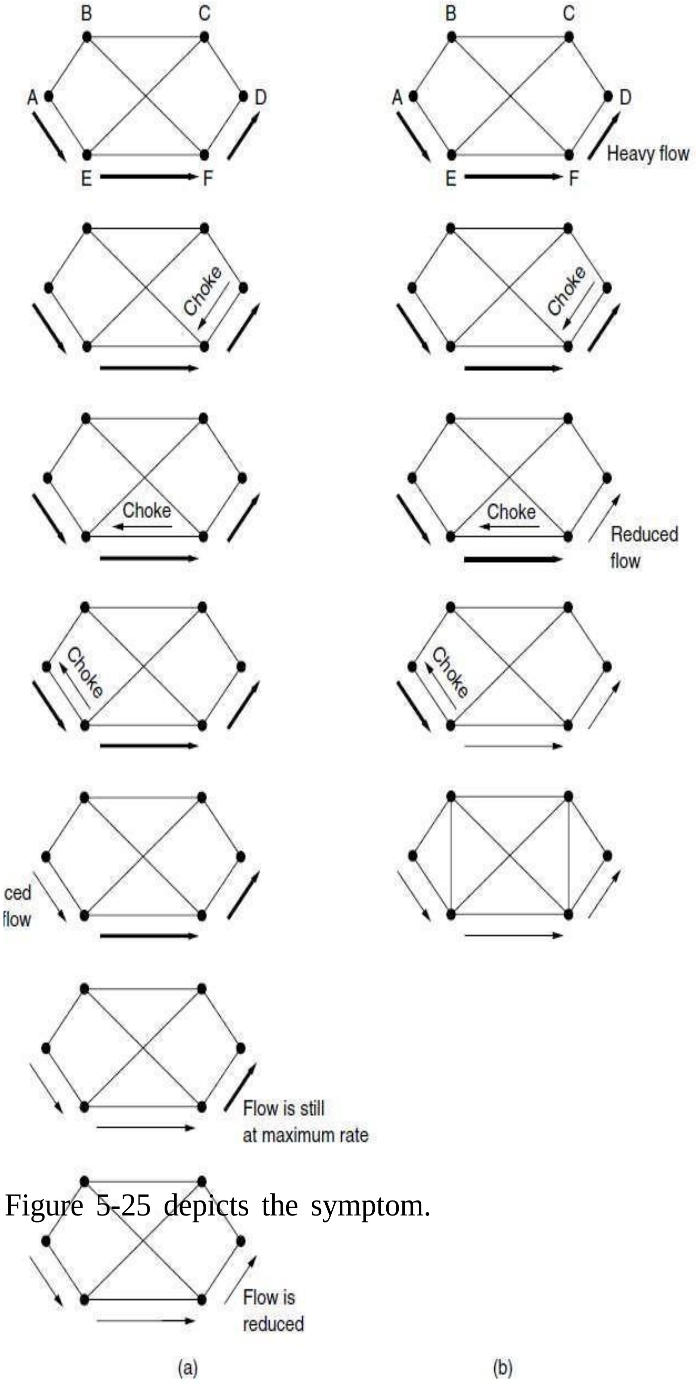
Explicit Congestion Notification

Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion. When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet. The sender can then throttle its transmissions as before. This design is called **ECN** (**Explicit Congestion Notification**). Packets are unmarked when they are sent, as illustrated in Fig. If any of the routers they pass through is congested, that router will then mark the packet as having experienced congestion as it is forwarded. The destination will then echo any marks back to the sender as an explicit congestion signal in its next reply packet.



Hop-by-Hop Backpressure

At high speeds or over long distances, many new packets may be transmitted after congestion has been signalled because of the delay before the signal takes effect. Consider, for example, a host in San Francisco (router *A* in Fig.) that is sending traffic to a host in New York (router *D* in Fig.) at the speed of 155 Mbps. If the New York host begins to run out of buffers, it will take about 40 msec for a choke packet to get back to San Francisco to tell it to slow down. An ECN indication will take even longer because it is delivered via the destination.



Choke packet propagation is illustrated as the second, third, and fourth steps in Fig(a). In those 40 msec, another 6.2 megabits will have been sent. Even if the host in San Francisco completely shuts down immediately, the 6.2 megabits in the pipe will continue to pour in and have to be dealt with. Only in the seventh diagram in Fig(a) will the New York router notice a slower flow. An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig.(b). Here, as soon as the choke packet reaches *F*, *F* is required to reduce the flow to *D*. Doing so will require *F* to devote more buffers to the connection, since the source is still sending away at full blast, but it gives *D* immediate relief. In the next step, the choke packet reaches *E*, which tells *E* to reduce the flow to *F*. This action puts a greater demand on *E*'s buffers but gives *F* immediate relief. Finally, the choke packet reaches *A* and the flow genuinely slows down. The net effect of this hop-by-hop scheme is to provide quick relief at the point of congestion, at the price of using up more buffers upstream. In this way, congestion can be nipped in the bud without losing any packets.

Figure 5-25 depicts the symptom.

Load Shedding

Load shedding is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just throw them away. The key question for a router drowning in packets is which packets to drop. The preferred choice may depend on the type of applications that use the network. For a file transfer, an old packet is worth more than a new one. This is because dropping packet 6 and keeping packets 7 through 10, for example, will only force the receiver to do more work to buffer data that it cannot yet use. In contrast, for real-time media, a new packet is worth more than an old one. This is because packets become useless if they are delayed and miss the time at which they must be played out to the user.

Random Early Detection

A popular algorithm for doing this is called **RED** (**Random Early Detection**). To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some link exceeds a threshold, the link is said to be congested and a small fraction of the packets are dropped at random. Picking packets at random makes it more likely that the fastest senders will see a packet drop; this is the best option since the router cannot tell which source is causing the most trouble in a datagram network. The affected sender will notice the loss when there is no acknowledgement, and then the transport protocol will slow down. The lost packet is thus delivering the same message as a choke packet, but implicitly, without the router sending any explicit signal. RED routers improve performance compared to routers that drop packets only when their buffers are full, though they may require tuning to work well. For example, the ideal number of packets to drop depends on how many senders need to be notified of congestion. However, ECN is the preferred option if it is available. It works in exactly the same manner, but delivers a congestion signal explicitly rather than as a loss; RED is used when hosts cannot receive explicit signals.

QUALITY OF SERVICE:

An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. The name for this solution is **over provisioning**. The resulting network will carry application traffic without significant loss and, assuming a decent routing scheme, will deliver packets with low latency. Performance doesn’t get any better than thi s.

To some extent, the telephone system is over provisioned because it is rare to pick up a telephone and not get a dial tone instantly. There is simply so much capacity available that demand can almost always be met. The trouble with this solution is that it is expensive.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network?
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

No single technique deals efficiently with all these issues. Instead, a variety of techniques have been developed for use at the network (and transport) layer. Practical quality-of-service solutions combine multiple techniques. To this end, we will describe two versions of quality of service for the Internet called Integrated Services and Differentiated Services.

APPLICATION REQUIREMENTS:

A stream of packets from a source to a destination is called a **flow.** A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network. The needs of each flow can be characterized by four primary parameters: ***bandwidth***, ***delay***, ***jitter***, and ***loss***. Together, these determine the **QoS** (**Quality of Service**) the flow requires.

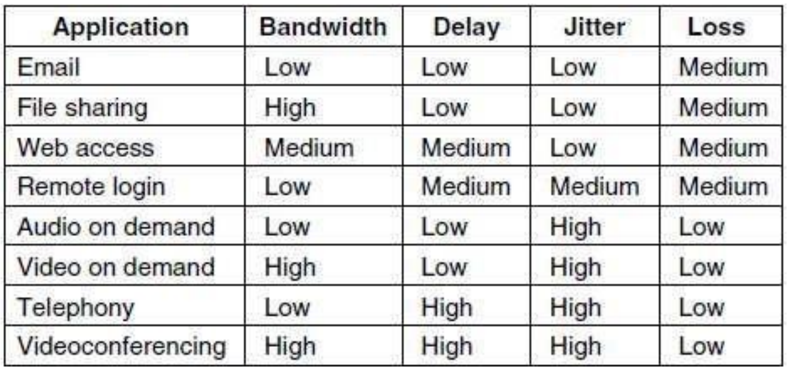
Several common applications and the stringency (meaning toughness/flexibility) of their network requirements are listed in Fig. The applications differ in their bandwidth needs, with email, audio in all forms, and remote login not needing much, but file sharing and video in all forms needing a great deal.

More interesting are the delay requirements. File transfer applications, including email and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done.

Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing, have strict delay requirements. If all the words in a telephone call are each delayed by too long, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

The variation (i.e., standard deviation) in the delay or packet arrival times is called **jitter**. The first three applications in Fig. are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat sensitive to that, since updates on the screen will appear in little bursts if the connection suffers much jitter.

Video and especially audio are extremely sensitive to jitter. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible unless the application hides the jitter. For audio, a jitter of even a few milliseconds is clearly audible.



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

FIGURE: STRINGENCY OF APPLICATIONS' QUALITY-OF-SERVICE REQUIREMENTS

To accommodate a variety of applications, networks may support different categories of QoS. An influential example comes from ATM networks. They support:

1. Constant bit rate (e.g., telephony).
2. Real-time variable bit rate (e.g., compressed videoconferencin g).
3. Non-real-time variable bit rate (e.g., watching a movie on demand).
4. Available bit rate (e.g., file transfer).

These categories are also useful for other purposes and other networks.

TRAFFIC SHAPING: Before the network can make QoS guarantees, it must know what traffic is being guaranteed. In the telephone network, this characterization is simple. For example, a voice call (in uncompressed format) needs 64 kbps and consists of one 8-bit sample every 125 μsec.

However, traffic in data networks is **bursty**. It typically arrives at nonuniform rates as the traffic rate varies (e.g., videoconferencing with compression), users interact with applications (e.g., browsing a new Web page), and computers switch between tasks. Bursts of traffic are more difficult to handle than constant-rate traffic because they can fill buffers and cause packets to be lost.

Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network. The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network.

When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow. In effect, the customer says to the provider ‘‘m y transmission pattern will look like this; can you handle it?’’

Sometimes this agreement is called an **SLA** (**Service Level Agreement**), especially when it is made over aggregate flows and long periods of time, such as all of the traffic for a given customer. As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the provider promises to deliver them all in a timely fashion.

Traffic shaping reduces congestion and thus helps the network live up to its promise. However, to make it work, there is also the issue of how the provider can tell if the customer is following the agreement and what to do if the customer is not. Packets in excess of the agreed pattern might be dropped by the network, or they might be marked as having lower priority. Monitoring a traffic flow is called **traffic policing**.

PACKET SCHEDULING:

Being able to regulate the shape of the offered traffic is a good start. However, to provide a performance guarantee, we must reserve sufficient resources along the route that the packets take through the network. To do this, we are assuming that the packets of a flow follow the same rout e. Spraying them over routers at random makes it hard to guarantee anything. As a consequence, something similar to a virtual circuit has to be set up from the source to the destination, and all the packets that belong to the flow must follow this route.

Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**. Three different kinds of resources can potentially be reserved for different flows:

1. Bandwidth.
2. Buffer space.
3. CPU cycles.

The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.

A second resource that is often in short supply is buffer space. When a packet arrives, it is buffered inside the router until it can be transmitted on the chosen outgoing line. The purpose of the buffer is to absorb small bursts of traffic as the flows contend with each other.

If no buffer is available, the packet has to be discarded since there is no place to put it. For good quality of service, some buffers might be reserved for a specific flow so that flow does not have to compete for buffers with other flows. Up to some maximum value, there will always be a buffer available when the flow needs one.

Finally, CPU cycles may also be a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. While modern routers are able to process most packets quickly, some kinds of packets require greater CPU processing, such as the ICMP packets. Making sure that the CPU is not overloaded is needed to ensure timely processing of these packets.

How Networks Can Be Connected

Networks can be interconnected by different devices.

- In the ***physical layer***, networks can be connected by *repeaters* or *hubs*, which just move the bits from one network to an identical network. These are mostly analog devices and do not understand anything about digital protocols.
- In the ***Data Link Layer***, networks are connected by *bridges* and *switches.* They can accept frames, examine the MAC addresses, and forward the frames to a different network while doing minor protocol translation in the process.
- In the ***network layer***, we have *routers* that can connect two networks. If two networks have dissimilar network layers, the router may be able to translate between

the packet formats, although packet translation is now increasingly rare. A router that can handle multiple protocols is called a **multiprotocol router**.

- In the ***transport layer*** we find transport gateways, which can interface between two transport connections. Transport gateway has a different transport protocol, by essentially gluing one connection to another connection.
- Finally, in the application layer, application gateways translate message semantics.

Here, we will focus on internetworking in the network layer. To see how that differs from switching in the data link layer, examine Fig. In Fig.(a), the source machine, *S*, wants to send a packet to the destination machine, *D*. These machines are on different Ethernets, connected by a switch. *S* encapsulates the packet in a frame and sends it on its way. The frame arrives at the switch, which then determines that the frame has to go to LAN 2 by looking at its MAC address. The switch just removes the frame from LAN 1 and deposits it on LAN 2.

INTERNETWORKING:

HOW NETWORKS DIFFER:

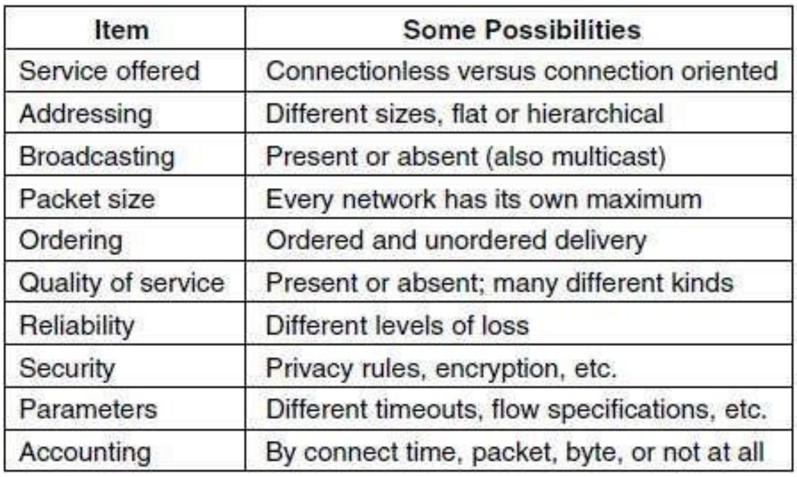
Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are internal to the physical and data link layers. These differences will not concern us here. Instead, in Fig. 3.12 we list some of the differences that can be exposed to the network layer. It is papering over these differences that makes internetworking more difficult than operating within a single network.

When packets sent by a source on one network must transit one or more foreign networks before reaching the destination network, many problems can occur at the interfaces between networks. To start with, the source needs to be able to address th e destination.

What do we do if the source is on an Ethernet network and the destination is on a WiMAX network? Assuming we can even specify a WiMAX destination from an Ethernet network, packets would cross from a connectionless network to a connection-oriented one.

This may require that a new connection be set up on short notice, which injects a delay, and much overhead if the connection is not used for many more packets. Many specific differences may have to be accommodated as well. How do we multicast a packet to a group with some members on a network that does not support multicast?

The differing max packet sizes used by different networks can be a major nuisance, too. How do you pass an 8000-byte packet through a network whose maximum size is 1500 bytes? If packets on a connection-oriented network transit a connectionless network, they may arrive in a different order than they were sent. That is something the sender likely did not expect, and it might come as an (unpleasant) surprise to the receiver as well.



Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

FIGURE : SOME OF THE MANY WAYS NETWORKS CAN DIFFER.

How Networks Can Be Connected

There are two basic choices for connecting different networks: we can build devices that translate or convert packets from each kind of network into packets for each other network, or, like good computer scientists, we can try to solve the problem by adding a layer of indirection and building a common layer on top of the different networks. In either case, the devices are placed at the boundaries between networks.

Internetworking has been very successful at building large networks, but it only works when there is a common network layer. There have, in fact, been many network protocols over time. Getting everybody to agree on a single format is difficult when companies perceive it to their commercial advantage to have a proprietary format that they control.

A router that can handle multiple network protocols is called a **multiprotocol router**. It must either translate the protocols, or leave connection for a higher protocol layer. Neither approach is entirely satisfactory. Connection at a higher layer, say, by using TCP, requires that all the networks implement TCP (which may not be the case). Then, it limits usage across the networks to applications that use TCP (which does not include many real-time applications).

TUNNELING:

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable even for different network protocols. This case is where the source and destination hosts are on the same type of network, but there is a different network in between. As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig.

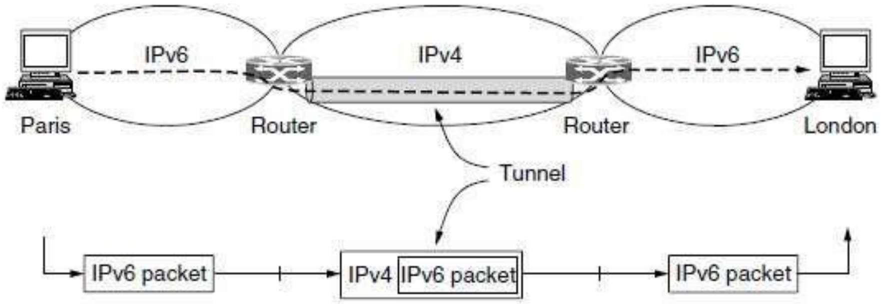


FIGURE: TUNNELING A PACKET FROM PARIS TO LONDON

The solution to this problem is a technique called **tunneling**. To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet.

When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network.

That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host. The path through the IPv4 Internet can be seen as a big tunnel extending from one multiprotocol router to the other.

The IPv6 packet just travels from one end of the tunnel to the other, snug in its nice box. It does not have to worry about dealing with IPv4 at all. Neither do the hosts in Paris or London. Only the multiprotocol routers have to understand both IPv4 and IPv6 packets.

In effect, the entire trip from one multiprotocol router to the other is like a hop over a single link. Tunneling is widely used to connect isolated hosts and networks using other networks.

INTERNETWORK ROUTING:

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet.

Networks run by different operators lead to bigger problems. First, the operators may have different ideas about what is a good path through the network. One operator may want the route with the least delay, while another may want the most inexpensive route. This will lead the operators to use different quantities to set the shortest-path costs.

Finally, the internet may be much larger than any of the networks that comprise it. It may therefore require routing algorithms that scale well by using a hierarchy, even if none of the individual networks need to use a hierarchy.

All of these considerations lead to a two-level routing algorithm. Within each network, an **intradomain** or **interior gateway protocol** is used for routing. (‘‘Gateway’’ is an older term for ‘‘router.’’) It might be a link state protocol of the Kind.

Across the networks that make up the internet, an **interdomain** or **exterior gateway protocol** is used. The networks may all use different intradomain protocols, but they must use the same interdomain protocol.

*In the Internet, the interdomain routing protocol is called **BGP (Border Gateway Protocol)**.*

There is one more important term to introduce. Since each network is operated independently of all the others, it is often referred to as an **AS** (**Autonomous System**). A good mental model for an AS is an ISP network. In fact, an ISP network may be comprised of more than one AS, if it is managed, or, has been acquired, as multiple networks. But the difference is usually not significant.

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

1. Hardware (e.g., the size of an Ethernet frame).

2. Operating system (e.g., all buffers are 512 bytes).

3. Protocols (e.g., the number of bits in the packet length field).

4. Compliance with some (inter)national standard.

5. Desire to reduce error-induced retransmissions to some level.

6. Desire to prevent one packet from occupying the channel too long.

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

Hosts usually prefer to transmit large packets because this reduces packet overheads such as bandwidth wasted on header bytes. An obvious internetworking problem appears when a large packet wants to travel through a network whose maximum packet size is too small. This nuisance has been a persistent issue, and solutions to it have evolved along with much experience gained on the Internet.

One solution is to make sure the problem does not occur in the first place. However, this is easier said than done. A source does not usually know the path a packet will take through the network to a destination, so it certainly does not know how small packets must be to get there. This packet size is called the **Path MTU** (**Path Maximum Transmission Unit**).

The alternative solution to the problem is to allow routers to break up packets into **fragments**, sending each fragment as a separate network layer packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process.

THE NETWORK LAYER IN THE INTERNET

THE IP VERSION 4 PROTOCOL:

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 3.14. The bits are transmitted from left to right and top to bottom, with the high-order bit of the *Version* field going first. (This is a ‘‘big-endian’’ network byte order.

On little-endian machines, such as Intel x86 computers, a software conversion is required on both transmission and reception.) In retrospect, little-endian would have been a better choice, but at the time IP was designed, no one knew it would come to dominate computing.

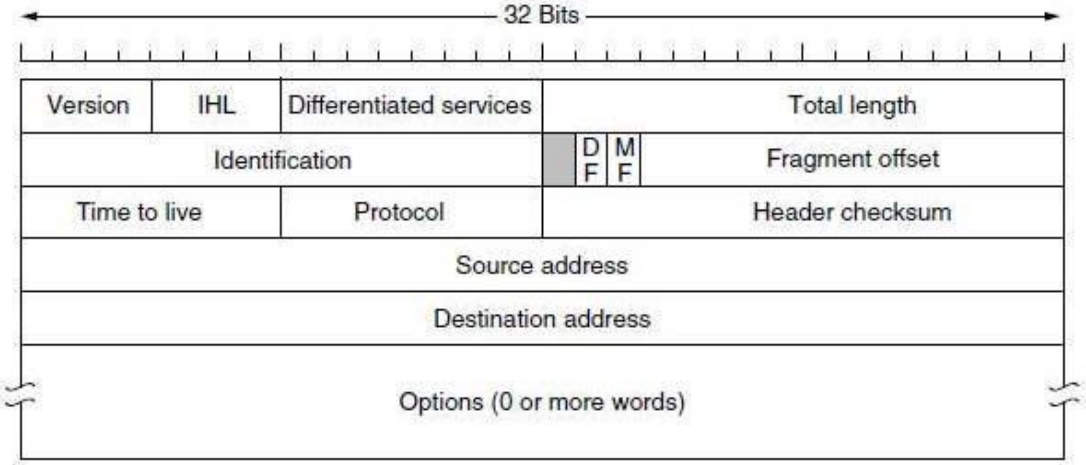


FIGURE: THE IPV4 (INTERNET PROTOCOL) HEADER

The *Version* field keeps track of which version of the protocol the datagram belongs to.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes.

The *Differentiated services* field is one of the few fields that have changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery.

For file transfer, error-free transmission is more important than fast transmission. The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability.

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

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

DF stands for Don’t Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again.

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

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

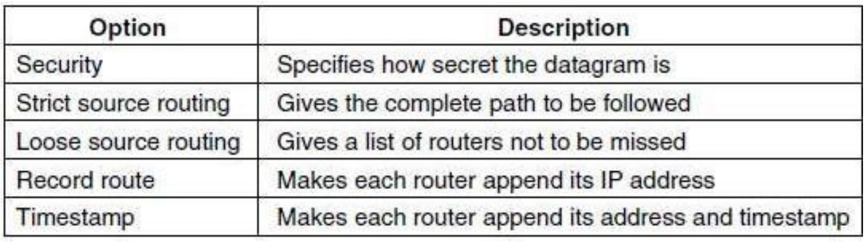
The *TtL (Time to live)* field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network.

The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces.

The *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. The *Options* field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig



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

FIGURE: SOME OF THE IP OPTIONS

IPV4 ADDRESSES:

The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address. An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is the address of the connection, not the host or the router, because if the device is moved to another network, the IP address may be changed.

IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses. IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space

A protocol like IPv4 that defines addresses has an address space. An **address space** is the total number of addresses used by the protocol. If a protocol uses *b* bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1). IPv4 uses 32-bit addresses, which means that the address space is 232 or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). In *binary notation,* an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces are usually inserted between each octet (8 bits). Each octet is often referred to as a byte. To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes.

This format is referred to as *dotted-decimal notation*. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255. We sometimes see an IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming. Figure 3.16 shows an IP address in the three discussed notations.

HIERARCHY IN ADDRESSING: A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the *prefix*, defines the network; the second part of the address, called the *suffix*, defines the node (connection of a device to the Inter net).

Figure shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is *n* bits and the suffix length is (32 − *n*) bits.

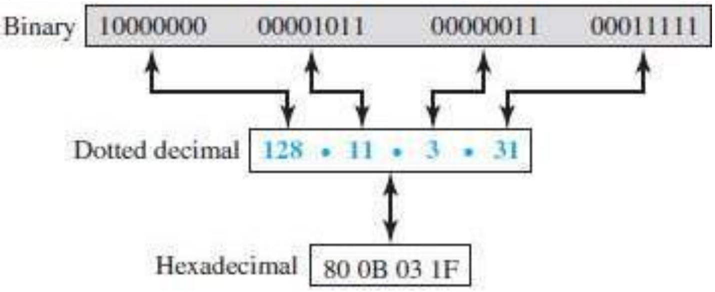


FIGURE 3.16: THREE DIFFERENT NOTATIONS IN IPV4 ADDRESSING

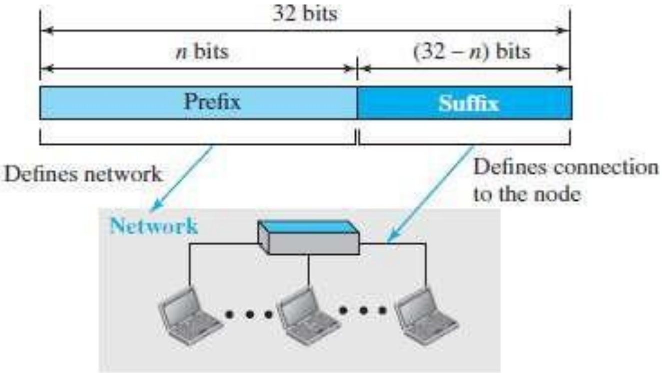


FIGURE : HIERARCHY IN ADDRESSING

A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing. The new scheme, which is referred to as classless addressing, uses a variable-length network prefix. First, we briefly discuss Classful addressing; then we concentrate on classless addressing.

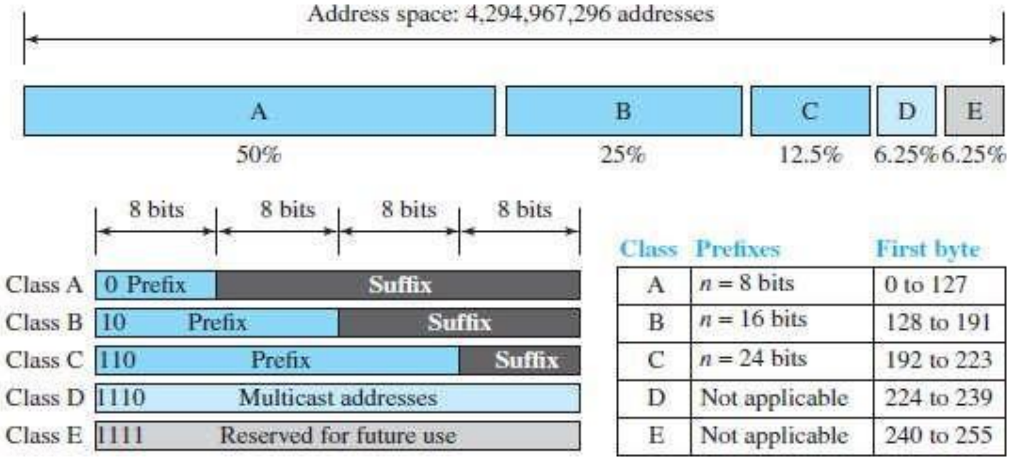
Classful Addressing:

When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one (*n* = 8, *n* = 16, and *n* = 24). The whole address space was divided into five classes (class A, B, C, D, and E), as shown in Figure. This scheme is referred to as **classful addressing.**

In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only 27 = 128 networks in the world that can have a class A address.

In class B, the network length is 16 bits, but since the first two bits, which are (10)2, define the class, we can have only 14 bits as the network identifier. This means there are only 214 = 16,384 networks in the world that can have a class B address.

All addresses that start with (110)2 belong to class C. In class C, the network length is 24 bits, but since three bits define the class, we can have only 21 bits as the network identifier. This means there are 221 = 2,097,152 networks in the world that can have a class C address.



Class	Prefixes	First byte
A	n = 8 bits	0 to 127
B	n = 16 bits	128 to 191
C	n = 24 bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

FIGURE: OCCUPATION OF THE ADDRESS SPACE IN CLASSFUL ADDRESSING

Class D is not divided into prefix and suffix. It is used for multicast addresses. All addresses that start with 1111 in binary belong to class E. As in Class D, Class E is not divided into prefix and suffix and is used as reserve.

Advantage of Classful Addressing:

Although classful addressing had several problems and became obsolete, it had one advantage: Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately. In other words, the prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

Address Depletion: The reason that classful addressing has become obsolete is address depletion. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet.

Subnetting and Supernetting: To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and Supernetting. In subnetting, a class A or class B block is divided into several subnets.

Each subnet has a larger prefix length than the original network. While subnetting was devised to divide a large block into smaller ones, Supernetting was devised to combine several class C blocks into a larger block to be attractive to organizations that need more than the 256 addresses available in a class C block. This idea did not work either because it makes the routing of packets more difficult.

Classless Addressing:

Subnetting and Supernetting in classful addressing did not really solve the address depletion problem. With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution. The larger address space, however, requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed.

Although the long-range solution has already been devised and is called IPv6, a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization. The short-term solution still uses IPv4 addresses, but it is called *classless addressing*. In other words, the class privilege was removed from the distribution to compensate for the address depletion.

In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network); the suffix defines the node (device). Theoretically, we can have a block of 20, 21, 22, . . . , 232 addresses. One of the restrictions, as we discuss later, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure shows the division of the whole address space into nonoverlapping blocks.

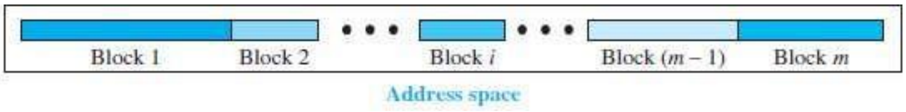


FIGURE: VARIABLE-LENGTH BLOCKS IN CLASSLESS ADDRESSING

Unlike classful addressing, the prefix length in classless addressing is variable. We can have a prefix length that ranges from 0 to 32. The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

We need to emphasize that the idea of classless addressing can be easily applied to classful addressing. An address in class A can be thought of as a classless address in which the prefix length is 8. An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

Prefix Length: Slash Notation:

The first question that we need to answer in classless addressing is how to find the prefix length if an address is given. Since the prefix length is not inherent in the address, we need to separately give the length of the prefix. In this case, the prefix length, *n*, is added to the address, separated by a slash. The notation is informally referred to as *slash notation* and formally as ***classless interdomain routing*** or ***CIDR*** (pronounced cider) strategy. An address in classless addressing can then be represented as shown.

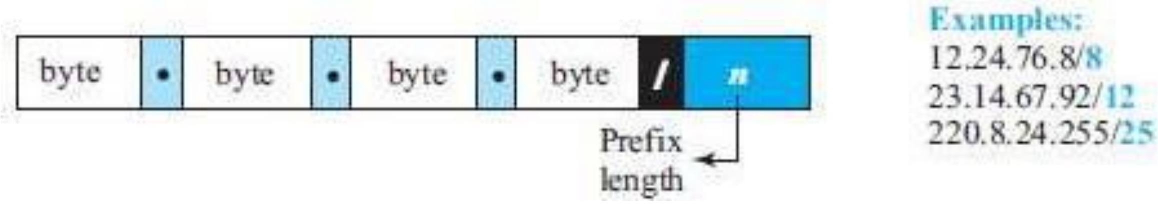


FIGURE: SLASH NOTATION (CIDR)

Extracting Information from an Address:

Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs: the number of addresses, the first address in the block, and the last address. Since the value of prefix length, *n*, is given, we can easily find these three pieces of information, as shown in Figure.

1. The number of addresses in the block is found as $N = 2^{32-n}$.
2. To find the first address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 0s.
3. To find the last address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 1s.

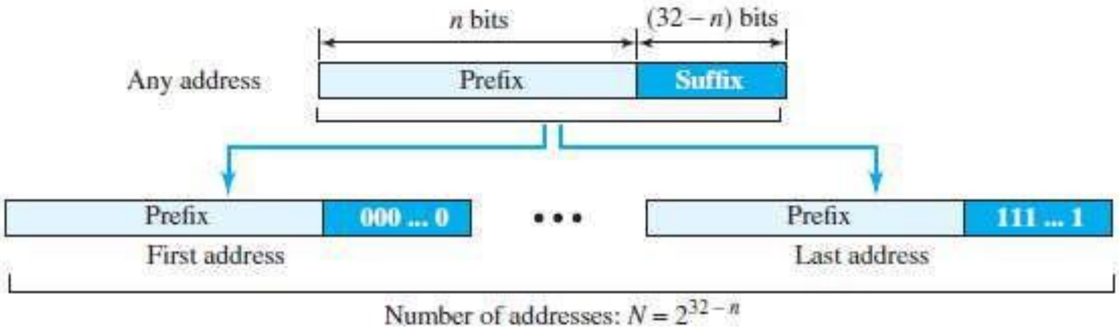


FIGURE: INFORMATION EXTRACTION IN CLASSLESS ADDRESSING

Example:

A classless address is given as 167.199.170.82/**27**. We can find the above three pieces of information as follows. The number of addresses in the network is 2^{32-n} = 25 = 32 addresses.

The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/**27** **10100111 11000111 10101010 01010010**
First address: 167.199.170.64/**27** **10100111 11000111 10101010 01000000**

The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/**27** **10100111 11000111 10101010 01011111**
Last address: 167.199.170.95/**27** **10100111 11000111 10101010 01011111**

IP VERSION 6:

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses. Even with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012.

IPv6 (**IP version 6**) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case.

In 1990 IETF started work on a new version of IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

1. Support billions of hosts, even with inefficient address allocation.

2. Reduce the size of the routing tables.

3. Simplify the protocol, to allow routers to process packets faster.

4. Provide better security (authentication and privacy).

5. Pay more attention to the type of service, particularly for real-time data.

6. Aid multicasting by allowing scopes to be specified.

7. Make it possible for a host to roam without changing its address.

8. Allow the protocol to evolve in the future.

9. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever.

IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses.

The main features of IPv6 are discussed below.

- First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses.

- The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay.

- The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often).

    - In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

- A fourth area in which IPv6 represents a big advance is in security.

- Finally, more attention has been paid to quality of service.

The Main IPv6 Header:

The IPv6 header is shown in Fig. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have.

As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for demultiplexing, so some routers may skip the check.

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

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo connection.

The *Payload length* field tells how many bytes follow the 40-byte header . The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

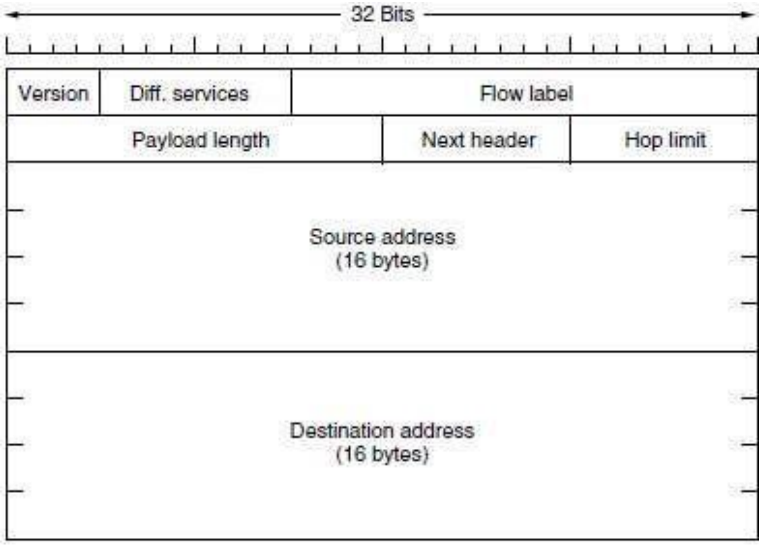


FIGURE 3.22: THE IPV6 FIXED HEADER (REQUIRED)

The *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to.

The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In

Next come the *Source address* and *Destination address* fields. A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes

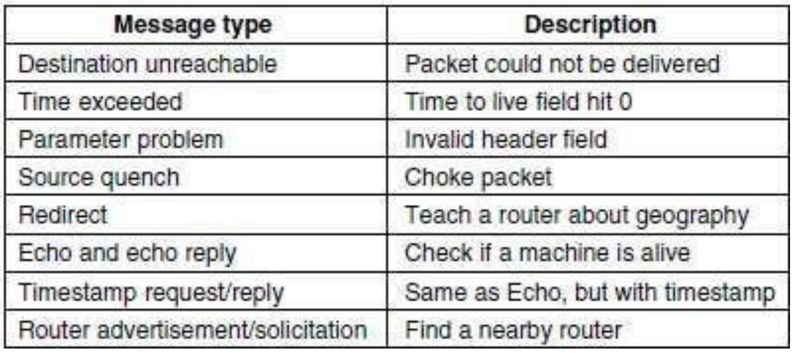
8000::123:4567:89AB:CDEF

INTERNET CONTROL PROTOCOLS:

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP.

ICMP—The Internet Control Message Protocol:

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP** (**Internet Control Message Protocol**). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined. Each ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig.



Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

FIGURE: THE PRINCIPAL ICMP MESSAGE TYPES

The DESTINATION UNREACHABLE message is used when the router cannot locate the destination or when a packet with the *DF* bit cannot be delivered because a ‘‘small-packet’’ network stands in the way.

The TIME EXCEEDED message is sent when a packet is dropped because its *TtL (Time to live)* counter has reached zero. This event is a symptom that packets are looping, or that the counter values are being set too low.

The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in the software of a router transited.

The SOURCE QUENCH message was long ago used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow do wn.

The REDIRECT message is used when a router notices that a packet seems to be routed incorrectly. It is used by the router to tell the sending host to update to a better route.

The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility can be used to measure network performance.

OSPF—AN INTERIOR GATEWAY ROUTING PROTOCOL:

The Internet is made up of a large number of independent networks or **ASes** (**Autonomous Systems**) that are operated by different organizations, usually a company, university, or ISP. Inside of its own network, an organization can use its own algorithm for internal routing, or **intradomain routing**, as it is more commonly known. Nevertheless, there are only a handful of standard protocols that are popular.

An intradomain routing protocol is also called an **interior gateway protocol**. We will study the problem of routing between independently operated networks, or **interdomain routing**. For that case, all networks must use the same interdomain routing protocol or **exterior gateway protocol**. The protocol that is used in the Internet is BGP (Border Gateway Protocol).

Early intradomain routing protocols used a distance vector design, based on the distributed Bellman-Ford algorithm inherited from the ARPANET. It works well in small systems, but less well as networks get larger. It also suffers from the count-to-infinity problem and generally slow convergence.

The ARPANET switched over to a link state protocol in May 1979 because of these problems, and in 1988 IETF began work on a link state protocol for intradomain routing. That protocol, called **OSPF** (**Open Shortest Path First**), became a standard in 1990. It drew on a protocol called **IS-IS** (**Intermediate-System to Intermediate-System**), which became an ISO standard.

Given the long experience with other routing protocols, the group designing OSPF had a long list of requirements that had to be met. First, the algorithm had to be published in the open literature, hence the ''O'' in OSPF.

Second, the new protocol had to support a variety of distance metrics, including physical distance, delay, and so on. Third, it had to be a dynamic algorithm, one that adapted to changes in the topology automatically and quickly.

Fourth, and new for OSPF, it had to support routing based on type of service. The new protocol had to be able to route real-time traffic one way and other traffic a different way. At the time, IP had a *Type of service* field, but no existing routing protocol used it. This field was included in OSPF but still nobody used it, and it was eventually removed.

Fifth, and related to the above, OSPF had to do load balancing, splitting the load over multiple lines. Most previous protocols sent all packets over a single best route, even if there were two routes that were equally good. The other route was not used at all. In many cases, splitting the load over multiple routes gives better performance.

Sixth, support for hierarchical systems was needed. By 1988, some networks had grown so large that no router could be expected to know the entire topology. OSPF had to be designed so that no router would have to.

OSPF supports both point-to-point links (e.g., SONET) and broadcast networks (e.g., most LANs). Actually, it is able to support networks with multiple routers, each of which can communicate directly with the others (called **multi-access networks**) even if they do not have broadcast capability. Earlier protocols did not handle this case well.

OSPF works by exchanging information between adjacent routers, which is not the same as between neighboring routers. In particular, it is inefficient to have every router on a LAN talk to every other router on the LAN. To avoid this situation, one router is elected as the **designated router**. It is said to be **adjacent** to all the other routers on its LAN, and exchanges information with them.

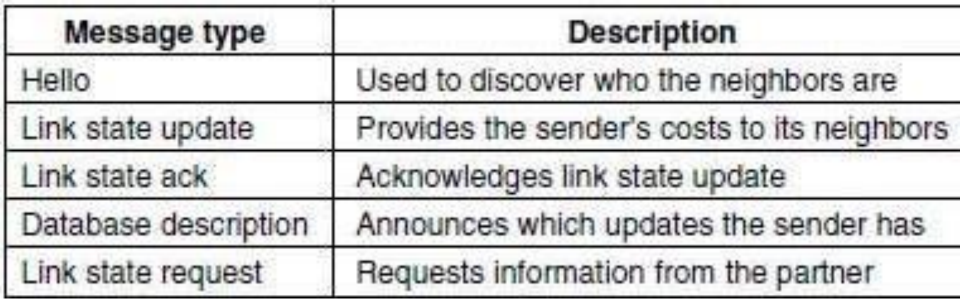
In effect, it is acting as the single node that represents the LAN. Neighboring routers that are not adjacent do not exchange information with each other. A backup designated router is always kept up to date to ease the transition should the primary designated router crash and need to be replaced immediately.

During normal operation, each router periodically floods LINK STATE UPDATE messages to each of its adjacent routers. These messages gives its state and provide the costs used in the topological database. The flooding messages are acknowledged, to make them reliable.

Each message has a sequence number, so a router can see whether an incoming LINK STATE UPDATE is older or newer than what it currently has. Routers also send these messages when a link goes up or down or its cost changes.

DATABASE DESCRIPTION messages give the sequence numbers of all the link state entries currently held by the sender. By comparing its own values with those of the sender, the receiver can determine who has the most recent values. These messages are used when a link is brought up.

All these messages are sent directly in IP packets. The five kinds of messages are summarized in Fig.



Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

FIGURE: THE FIVE TYPES OF OSPF MESSAGES

BGP—THE EXTERIOR GATEWAY ROUTING PROTOCOL:

Within a single AS, OSPF and IS-IS are the protocols that are commonly used. Between ASes, a different protocol, called **BGP** (**Border Gateway Protocol**), is used. A different protocol is needed because the goals of an intradomain protocol and an interdomain protocol are not the same. All an intradomain protocol has to do is move packets as efficiently as possible from the source to the destination.

BGP is a form of distance vector protocol, but it is quite unlike intradomain distance vector protocols such as RIP. We have already seen that policy, instead of minimum distance, is used to pick which routes to use. Another large difference is that instead of maintaining just the cost of the route to each destination, each BGP router keeps track of the path used. This approach is called a **path vector protocol**.

The path consists of the next hop router (which may be on the other side of the ISP, not adjacent) and the sequence of ASes, or **AS path**, that the route has followed (given in reverse order). Finally, pairs of BGP routers communicate with each other by establishing TCP connections. Operating this way provides reliable communication and also hides all the details of the network being passed through.

An example of how BGP routes are advertised is shown in Fig. 3.25. There are three ASes and the middle one is providing transit to the left and right ISPs. A route advertisement to prefix *C* starts in *AS3*. When it is propagated across the link to *R2c* at the top of the figure, it has the AS path of simply *AS3* and the next hop router of *R3a*.

At the bottom, it has the same AS path but a different next hop because it came across a different link. This advertisement continues to propagate and crosses the boundary into *AS1*. At router *R1a*, at the top of the figure, the AS path is *AS2*, *AS3* and the next hop is *R2a*.

Carrying the complete path with the route makes it easy for the receiving router to detect and break routing loops. The rule is that each router that sends a route outside of the AS prepends its own AS number to the route. (This is why the list is in reverse order.)

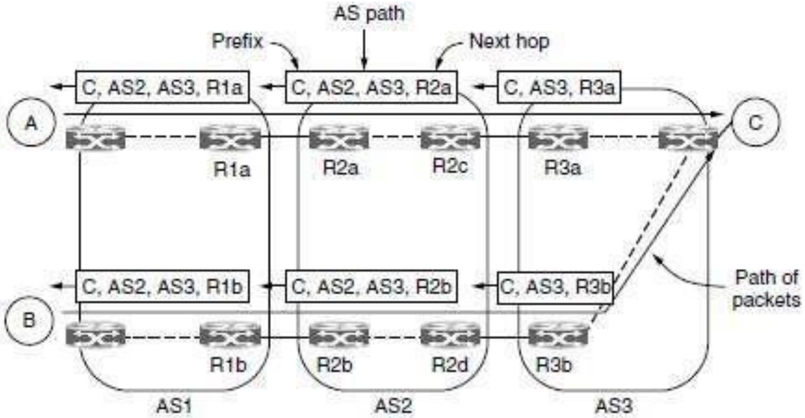


FIGURE : PROPAGATION OF BGP ROUTE ADVERTISEMENTS

When a router receives a route, it checks to see if its own AS number is already in the AS path. If it is, a loop has been detected and the advertisement is discarded.

INTERNET PROTOCOL (IP):

The network layer in version 4 can be thought of as one main protocol and three auxiliary ones. The main protocol, Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer.

The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery. The Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting. The Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses. Figure 3.26 shows the positions of these four protocols in the TCP/IP protocol suite.

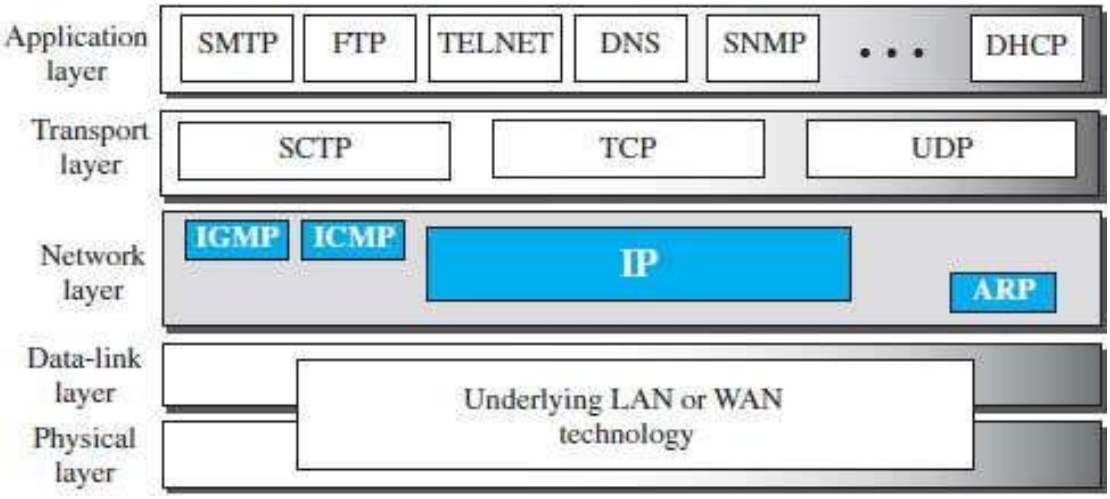


FIGURE : POSITION OF IP & OTHER NETWORK-LAYER PROTOCOLS IN TCP/IP PROTOCOL SUITE

IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term *best-effort* means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network. If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP.

IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Again, IPv4 relies on a higher -level protocol to take care of all these problems.

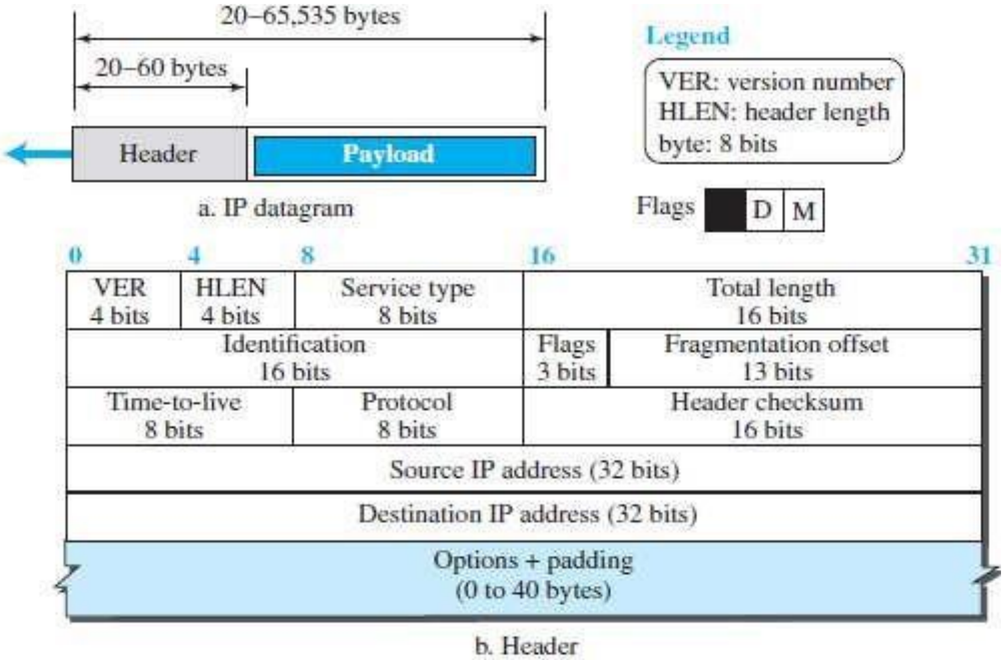
DATAGRAM FORMAT:

Packets used by the IP are called *datagrams*. Figure shows the IPv4 datagram format. A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.

- ***Version Number.*** The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.

- ***Header Length.*** The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header.

- ***Service Type.*** In the original design of the IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled.



a. IP datagram

b. Header

FIGURE : IP DATAGRAM

- ***Total Length.*** This 16-bit field defines the total length (header plus data) of the IP datagram in bytes. A 16-bit number can define a total length of up to 65,535 (when all bits are 1s).

- ***Identification, Flags, and Fragmentation Offset.*** These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry.

- ***Time-to-live.*** The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

- ***Protocol*****.** In TCP/IP, the data section of a packet, called the *payload*, carries the whole packet from another protocol. A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP. A datagram can also carry a packet from other protocols that directly use the service of the IP, such as some routing protocols or some auxiliary protocols.

- ***Header checksum*****.** IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission. IP puts the burden of error checking of the payload on the protocol that owns the payload, such as UDP or TCP. The datagram header, however, is added by IP, and its error-checking is the responsibility of IP.

- ***Source and Destination Addresses*****.** These 32-bit source and destination address fields define the IP address of the source and destination respectively. The source host should know its IP address. The destination IP address is either known by the protocol that uses the service of IP or is provided by the DNS.

- ***Options.*** A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software.

- ***Payload*****.** Payload, or data, is the main reason for creating a datagram. Payload is the packet coming from other protocols that use the service of IP. Comparing a datagram to a postal package, payload is the content of the package; the header is only the information written on the package.

ICMPv4:

The IPv4 has no error-reporting or error-correcting mechanism. The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network manager needs information from another host or router.

The **Internet Control Message Protocol version 4 (ICMPv4)** has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. ICMP itself is a network-layer protocol.

However, its messages are not passed directly to the data-link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer. When an IP datagram encapsulates an ICMP message, the value of the protocol field in the IP datagram is set to 1 to indicate that the IP payroll is an ICMP message.

MESSAGES:

ICMP messages are divided into two broad categories: ***error-reporting messages*** and ***query messages***.

The ***error-reporting messages*** report problems that a router or a host (destination) may encounter when it processes an IP packet.

The ***query messages***, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network and routers can help a node redirect its messages.

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all.

As Figure shows, the first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field (to be discussed later in the chapter). The rest of the header is specific for each message type.

The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of query.

Error Reporting Messages: Since IP is an unreliable protocol, one of the main responsibilities of ICMP is to report some errors that may occur during the processing of the IP datagram. ICMP does not correct errors, it simply reports them.

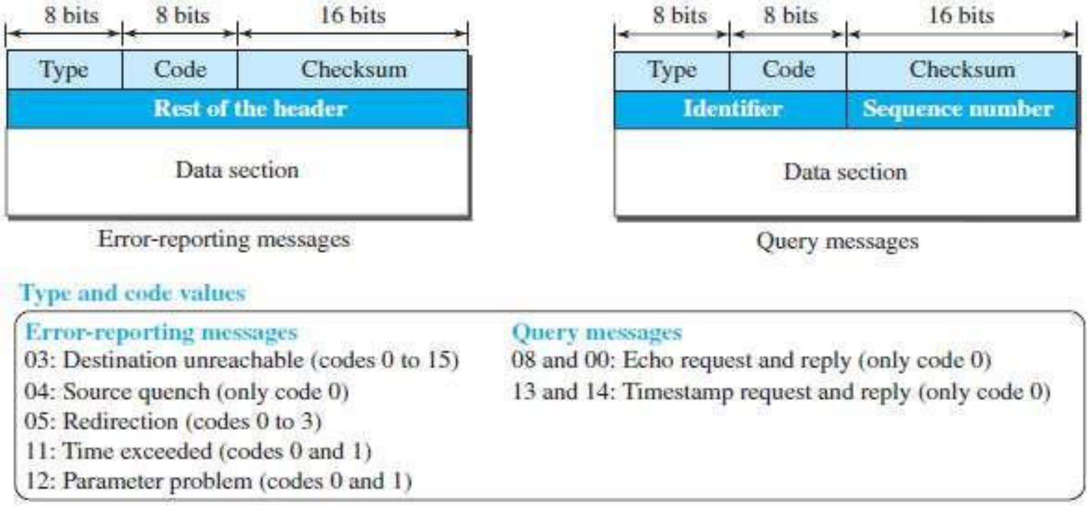
Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.

ICMP uses the source IP address to send the error message to the source (originator) of the datagram. To make the error-reporting process simple, ICMP follows some rules in reporting messages:

- First, no error message will be generated for a datagram having a multicast address or special address (such as *this host* or *loopback*).
- Second, no ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- Third, no ICMP error message will be generated for a fragmented datagram that is not the first fragment.

Destination Unreachable: The most widely used error message is the destination unreachable (type 3). This message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination.

Source Quench: Another error message is called the *source quench* (type 4) message, which informs the sender that the network has encountered congestion and the datagram has been dropped; the source needs to slow down sending more datagrams.



03: Destination unreachable (codes 0 to 15)
04: Source quench (only code 0)
05: Redirection (codes 0 to 3)
11: Time exceeded (codes 0 and 1)
12: Parameter problem (codes 0 and 1)

08 and 00: Echo request and reply (only code 0)
13 and 14: Timestamp request and reply (only code 0)

FIGURE: GENERAL FORMAT OF ICMP MESSAGES

Redirection Message: The *redirection message* (type 5) is used when the source uses a wrong router to send out its message. The router redirects the message to the appropriate router, but informs the source that it needs to change its default router in the future. The IP address of the default router is sent in the message.

Parameter Problem: A *parameter problem message* (type 12) can be sent when either there is a problem in the header of a datagram (code 0) or some options are missing or cannot be interpreted (code 1).

Query Messages: Query messages in ICMP can be used independently without relation to an IP datagram. Of course, a query message needs to be encapsulated in a datagram, as a carrier.

Query messages are used to probe or test the liveliness of hosts or routers in the Internet, find the one-way or the round-trip time for an IP datagram between two devices, or even find out whether the clocks in two devices are synchronized. Naturally, query messages come in pairs: request and reply.

IGMP:

The protocol that is used today for collecting information about group membership is the **Internet Group Management Protocol (IGMP).** IGMP is a protocol defined at the network layer; it is one of the auxiliary protocols, like ICMP, which is considered part of the IP. IGMP messages, like ICMP messages, are encapsulated in an IP datagram.

Messages:

There are only two types of messages in IGMP version 3, query and report messages, as shown in Figure. A query message is periodically sent by a router to all hosts attached to it to ask them to report their interests about membership in groups. A report message is sent by a host as a response to a query message.

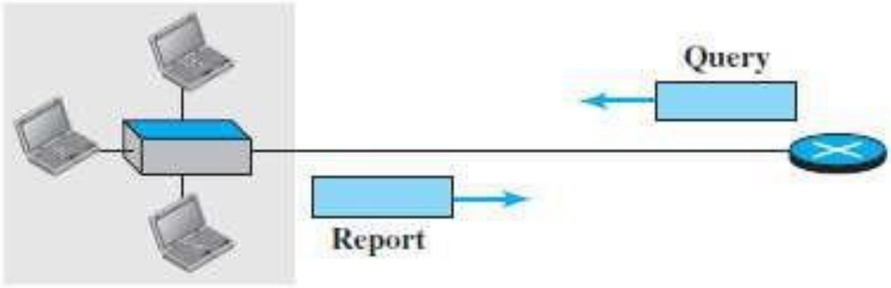


FIGURE: IGMP OPERATION

Query Message:

The query message is sent by a router to all hosts in each interface to collect information about their membership. There are three versions of query messages, as described below:

a. A ***general*** **query** message is sent about membership in any group. It is encapsulated in a datagram with the destination address 224.0.0.1 (all hosts and routers). Note that all routers attached to the same network receive this message to inform them that this message is already sent and that they should refrain from resending it.

b. A ***group-specific*** query message is sent from a router to ask about the membership related to a specific group. This is sent when a router does not receive a response about a specific group and wants to be sure that there is no active member of that group in the network. The group identifier (multicast address) is mentioned in the message. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

c.A ***source-and-group-specific*** query message is sent from a router to ask about the membership related to a specific group when the message comes from a specific source or sources. Again the message is sent when the router does not hear about a specific group related to a specific host or hosts. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

Report Message

A report message is sent by a host as a response to a query message. The message contains a list of records in which each record gives the identifier of the corresponding group (multicast address) and the addresses of all sources that the host is interested in receiving messages from (inclusion).

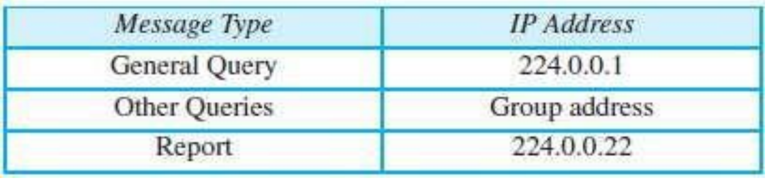
The record can also mention the source addresses from which the host does not desire to receive a group message (exclusion). The message is encapsulated in a datagram with the multicast address 224.0.0.22 (multicast address assigned to IGMPv3).

In IGMPv3, if a host needs to join a group, it waits until it receives a query message and then sends a report message. If a host needs to leave a group, it does not respond to a query message. If no other host responds to the corresponding message, the group is purged from the router database.

Propagation of Membership Information:

After a router has collected membership information from the hosts and other routers at its own level in the tree, it can propagate it to the router located in a higher level of the tree. Finally, the router at the tree root can get the membership information to build the multicast tree. The process, however, is more complex than what we can explain in one paragraph. Interested readers can check the book website for the complete description of this protocol.

Encapsulation: The IGMP message is encapsulated in an IP datagram with the value of the protocol field set to 2 and the TTL field set to 1. The destination IP address of the datagram, however, depends on the type of message, as shown in figure.



Message Type	*IP Address*
General Query	224.0.0.1
Other Queries	Group address
Report	224.0.0.22

FIGURE : DESTINATION IP ADDRESSES

UNIT-III

NETWORK LAYER

OBJECTIVE QUESTIONS

1. The network layer concerns with

a) bits b) frames

c) packets d) none of the mentioned

2. Which one of the following is not a function of network layer?

a) routing b) inter-networking

c) congestion control **d) none of the mentioned**

3. The 4 byte IP address consists of
a) network address b) host address
c) both (a) and (b) d) none

4. In virtual circuit network each packet contains
a) full source and destination address **b) a short VC number**
c) both (a) and (b) d) none

5. Which one of the following routing algorithm can be used for network layer design?
a) shortest path algorithm b) distance vector routing
c) link state routing **d) all the above**

6.6. Multidestination routing
a) is same as broadcast routing b) contains the list of all destinations
c) data is not sent by packets d) none

7. A subset of a network that includes all the routers but contains no loops is called
a) spanning tree b) spider structure
c) spider tree d) none

8. Which one of the following algorithm is not used for congestion control?
a) traffic aware routing b) admission control
c) load shedding **d) none of the mentioned**

9. The network layer protocol of internet is
a) Ethernet **b) internet protocol**
c) hypertext transfer protocol d) none of the mentioned

10. ICMP is primarily used for
a) error and diagnostic functions b) addressing
c) forwarding
d) none of the mentioned

11. The operation of subnet is controlled by ________ _.
a. Network Layer.
b. Data Link Layer
c. Data Layer
d. Transport Layer

12 are two popular examples of distance vector routing protocols.
a. OSPF and RIP **b. RIP and BGP**
c. BGP and OSPF d. BGP and SPF

13 deals with the issues of creating and maintaining routing tables.
a. Forwarding **b. Routing**
c. Directing d. None directing

14. During an adverse condition, the length of time for every device in the network to produce an accurate routing table is called the ______.

a. Accurate time b. integrated time

c. Convergence time d. Average time

15. A____ routing table contains information entered manually.

a. Static b. Dynamic

c. Hierarchical d. Non static

16. Which of the following is/are the uses of static routing methods?

a. To manually define a default route. b. To provide more secure network environment.

c. To provide more efficient resource utilization. **d. All of the above**

17. A____routing table is updated periodically using one of the dynamic routing protocols.

a. static **b. dynamic**

c. hierarchical d. non static

18.Which of the following is not the category of dynamic routing algorithm?

a. Distance vector protocols b. Link state protocols

c. Hybrid protocols **d. Automatic state protocols**

19. In ________forwarding, the full IP address of a destination is given in the routing table.

a. next-hop b. network-specific

c. host-specific d. default

20. To build the routing table, _________algorithms allow routers to automatically discover and maintain awareness or the paths through the network.

a. Static routing **b. dynamic routing**

c. Hybrid routing d. automatic routing

21. In ______ forwarding, the mask and destination addresses are both 0.0.0.0 in the routing table.

a. next-hop b. network-specific

c. host-specific **d. default**

22).To build the routing table, _______method use preprogrammed definitions representing paths through the network.

a. Static routing b. dynamic routing

c. Hybrid routing d. automatic routing

23).In _____ forwarding, the destination addresses is a network address in the routing table.

a. next-hop **b. network-specific**

c. host-specific d. default

24) ______allows routers to exchange information within an AS.

a. Interior Gateway Protocol (IGP) b. Exterior Gateway Protocol (EGP)

c. Border Gateway Protocol (BGP)   d. Static Gateway Protocol (SGP)

25. In ____ forwarding, the routing table holds the address of just the next hop instead of complete route information.
a. next-hop   b. network-specific
c. host-specific   d. default

26. Which of the following is an example of Exterior Gateway Protocol?
a. Open Short Path First (OSPF)   **b. Border Gateway Protocol (BGP)**
c. Routing Information Protocol (RIP)   d. All of the above

27. A one-to-all communication between one source and all hosts on a network is classified as a _.
a. unicast   b. multicast
c. broadcast   d. point to point

28.allow the exchange of summary information between autonomous systems.
a. Interior Gateway Protocol (IGP)   **b. Exterior Gateway Protocol (EGP)**
c. Border Gateway Protocol (BGP)   d. Dynamic Gateway Protocol (DGP)

29).A robust routing protocol provides the ability to.......... build and manage the information in the IP routing table.
a. Dynamically   b. Statically
c. Hierarchically   d. All of the above

30. State True of False for definition of an autonomous system(AS).
i) An AS is defined as a physical portion of a larger IP network.
ii) An AS is normally comprised of an internetwork within an organization.
a. i-True, ii-True   b. i-True, ii-False
c. i-False, ii-True   d. i-False, ii-False

31. What are the parameters on which two networks differ.
a) Packet sized used   b) use flow and error control technique
c) Connectionless control and security mechanism **d) all**

32 ________ are the limitations that cause different networks have different packet size.
a) hardware   b) operating system
c) protocols   **d) all**

33. Fragmentation means__________
a) adding of sma;ll packets to form large packets
b) breaking the large packet into small packets
c) forwarding packet through different networks
d) None

34. The header part of a fragment contains__________ number of fields
a) 2   **b)3**   c)1   d)4

35.The header checksum is the IP header is used to verify ________ _.
a) only header b) only data c) both d) None

36. The higest IPV4 address in digital notation is
a) 255.0.0.0 b)255.255.0.0 **c)255.255.255.255** d)255.255.255.255

37. Which class of IP address is used for used for more networks and less hosts.
a) class-A b) class-B **c)class-C** d) class-E

38. In IPV4 addressing, the digital notation address 222.255.255.255 belongs to
a) class-A b) class-B **c)class-C** d) class-E

39.classless inter domain routing contains___________ _.
a)32 bit IP address b)32 bit mask address **c)both** d)None

40.To indicate a sender that,it has no data,to the receiver bit is set
a)PSH b)RST **c)FIN** d) ACK

41.connecting different networks is called _____________
a) intranet working **b)internetworking** c)multiple networking d)ALL

42.Bridges are used in ________layer.
a) physical **b)MAC** c)network d)application

43.Which is a intranet working device
a) router b)gateway c)bridge **d)ALL**

44. Gateways are used at __________ layer.
a) datalink layer b)network layer **c)application** d)ALL

45.The max length of the option load field in IP datagram is ___________ _.
a)40 bytes b)80 bytes c)16 bytes d)any number of bytes multipleof 4

46. The length of the subnet mask is ______bits .
a)16 bits **b)32bits** c)64bits d)any

47.Address resolution protocol is used to MAP the IP address on to the_________.
a) data link layer b)internet address c)network address d)port address

48.RARP is used to map the data link layer address onto _______________address.
a)network b)port **c)IP** d)None

49.which are the following option are used in IPV4.
a) security b)timestamp c)source routing **d)ALL**

50.Which class of IP addressing provide more number of hosts in each network
a)class-A b)class-B c)class-c d)class-D

UNIT-III
Descriptive Questions

1. What are the Services provided by Network layer to the Transport layer ?

2. Discuss the functions of the communication subnet to provide datagram service.

3. What is meant by connection state information in a virtual circuit network?

4. Compare Virtual-Circuit and Datagram Subnets.

5. What is routing algorithm? What are the classifications of it?

6. What is the Optimality Principle?

7. With an example explain shortest path routing algorithm.

8. Explain flooding

9. Explain distance vector routing algorithm.

10. Explain count-to-infinity problem.
11. Write short notes on the following
    (a) IPV4 (b) IPV6
12. Write about Internet Control Protocols.

UNIT-IV

TRANSPORT LAYER AND

APPLICATION LAYER

The transport layer in the TCP/IP suite is located between the application layer and the network layer. It provides services to the application layer and receives services from the network layer. The transport layer acts as a liaison between a client program and a server program, a process-to-process connection. The transport layer is the heart of the TCP/IP protocol suite; it is the end-to-end logical vehicle for transferring data from one point to another in the Internet.

Introduction:

The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host. Communication is provided using a logical connection, which means that the two application layers, which can be located in different parts of the globe, assume that there is an imaginary direct connection through which they can send and receive messages.

THE TRANSPORT SERVICE:

Services provided to the upper layers:

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**.

The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 4.1. *Just as there are two types of network service*, ***connection-oriented*** *and* ***connectionless****, there are also two types of transport service*. The ***connection-oriented transport service*** is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.

Furthermore, the ***connectionless transport service*** is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear (*meaning run/rip/rush*) it down immediately afterwards.

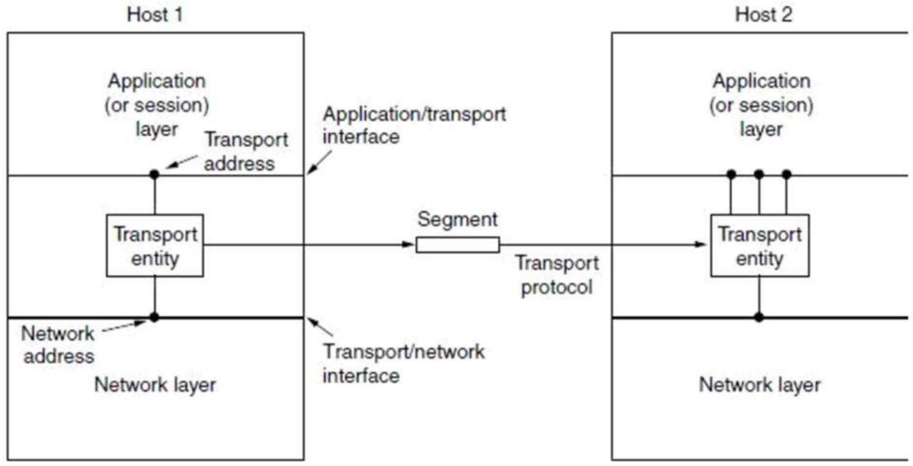


Figure 4.1: The network, transport, and application layers

Transport service primitives:

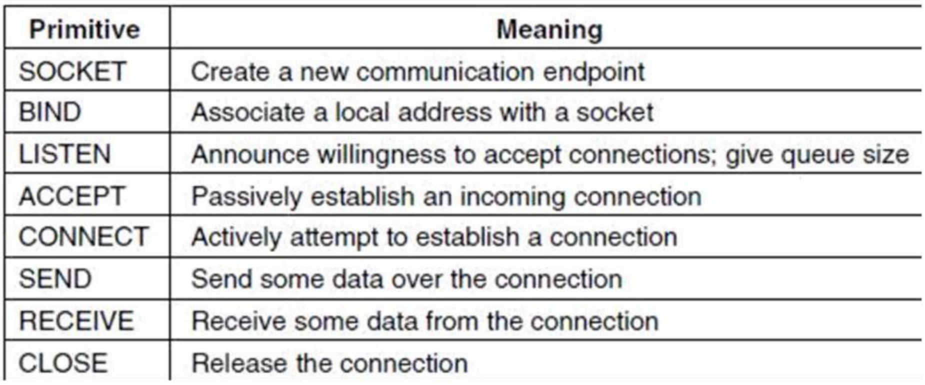
To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface. The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks and all. Real networks can lose packets, so the network service is generally unreliable. The connection-oriented transport service, in contrast, is reliable. Of course, real networks are not error-free, but that is precisely the purpose of the transport layer—to provide a reliable service on top of an unreliable network.

A second difference between the network service and transport service is whom the services are intended for. The network service is used only by the transport entities. Few users write their own transport entities, and thus few users or programs ever (*meaning always/forever/still*) see the bare network service.

Berkeley sockets:

Let us now briefly inspect another set of transport primitives, the socket primitives as they are used for TCP. Sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular.

The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called ‘‘winsock.’’ The primitives are listed in Fig. 4.2.



Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 4.2: The socket primitives for TCP

Note: An Example of Socket Programming: An Internet File Server

ELEMENTS OF TRANSPORT PROTOCOLS:

The transport service is implemented by a **transport protocol** used between the two transport entities. In some ways, transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues. However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 4.3.

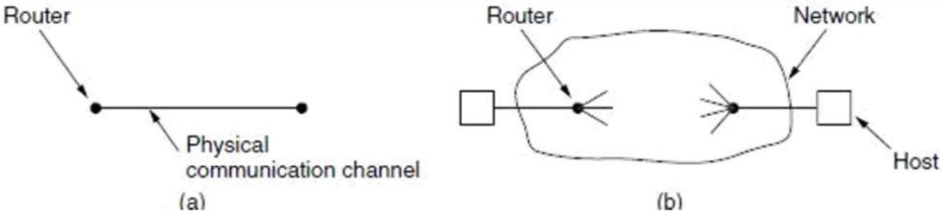


Figure 4.3: Environment of the (a) data link layer (b) transport layer

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 4.3(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do.

Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations. If the message is not acknowledged due to an error, it can be resent. In the transport layer, initial connection establishment is complicated.

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. (Connectionless transport has the same problem: to whom should each message be sent?) The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**. We will use the generic term **TSAP** (**Transport Service Access Point**) to mean a specific endpoint in the transport layer. The analogous endpoints in the network layer (i.e., network layer addresses) are naturally called **NSAPs** (**Network Service Access Points**). IP addresses are examples of NSAPs.

Figure 4.4 illustrates the relationship between the NSAPs, the TSAPs, and a transport connection.

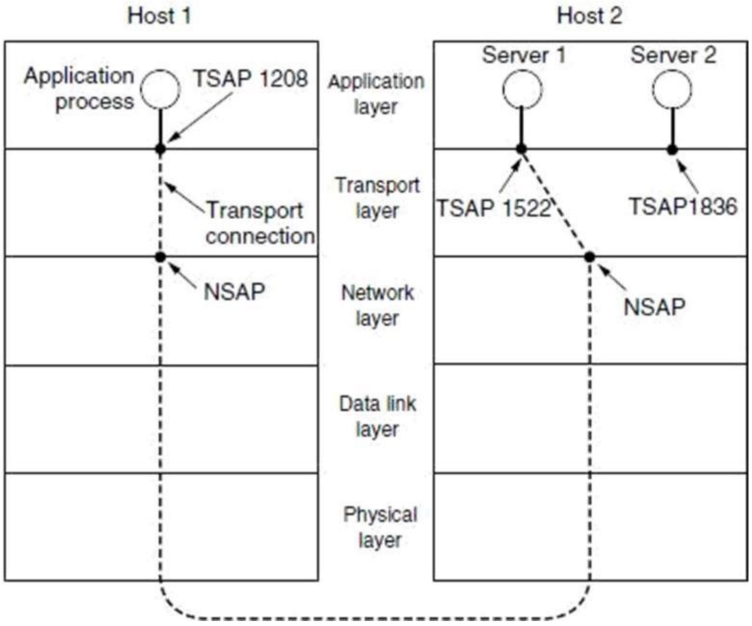


Figure 4.4: TSAPs, NSAPs, and Transport connections

Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host, as shown in figure 4.4.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. A call such as our LISTEN might be used, for example.

2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.

3. The application process sends over the mail message.

4. The mail server responds to say that it will deliver the message.

5. The transport connection is released.

Connection Establishment:

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications. Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route.

Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost. *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. Unfortunately, the packets decide to take the scenic route to the destination and go off exploring a remote corner of the network.*

The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

Unfortunately, eventually the initial batch of packets finally come out of hiding and arrive at the destination in order, asking the bank to establish a new connection and transfer money (again). The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

The crux (*meaning root*) of the problem is that the delayed duplicates are thought to be new packets. We cannot prevent packets from being duplicated and delayed. But if and when this happens, the packets must be rejected as duplicates and not processed as fresh packets. The problem can be attacked in various ways, none of them very satisfactory. 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. Delayed duplicate packets then never find their way to a transport process and can do no damage.

Note: However, this approach makes it more difficult to connect with a process in the first place.

Another possibility is to give each connection a unique identifier (i.e., a sequence number incremented for each connection established) chosen by the initiating party and put in each segment, including the one requesting the connection. After each connection is released, each transport entity can update a table listing obsolete connections as (peer transport entity, connection identifier) pairs. Whenever a connection request comes in, it can be checked against the table to see if it belongs 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. This history must persist at both the source and destination machines. Otherwise, if a machine crashes and loses its memory, it will no longer know which connection identifiers have already been used by its peers.

Instead, we need to take a different tack to simplify the problem. Rather than allowing packets to live forever within the network, we devise a mechanism to kill off aged packets that are still hobbling about.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

TCP uses three-way handshake to establish connections in the presence of delayed duplicate control segments as shown in figure 4.5.

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 party hangs up, the connection is broken. ***Symmetric release*** treats the connection as two separate unidirectional connections and requires each one to be released separately.

Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. 4.6. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment.

Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. In other situations, determining that all the work has been done and the connection should be terminated is not so obvious. One can envision a protocol in which host 1 says ‘‘I am done. Are you done too?’’ If host 2 responds: ‘‘I am done too. Goodbye, the connection can be safely released.’’

In practice, we can avoid this quandary (*meaning dilemma/difficulty*) by foregoing the need for agreement and pushing the problem up to the transport user, letting each side independently decide when it is done. This is an easier problem to solve. Figure 4.7 illustrates four scenarios of releasing using a three-way handshake. While this protocol is not infallible, it is usually adequate. In Fig. 4.7(a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release.

When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost. When this DR arrives, the original sender sends back an ACK segment and releases the connection.

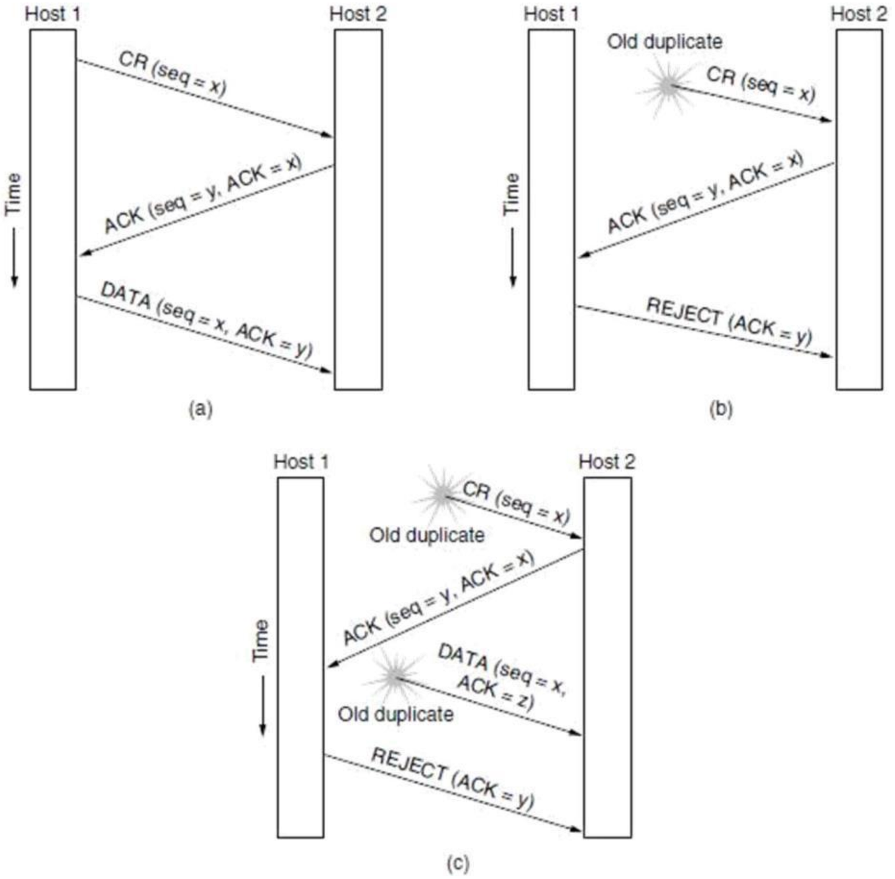


Figure 4.5: Three protocol scenarios for establishing a connection using a three - way handshake. CR denotes Connection Request. (a) normal operation. (b) old duplicate connection request appearing out of nowhere.
(c) duplicate connection request and duplicate ack.

Finally, when the ACK segment 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. If the final ACK segment is lost, as shown in Fig. 4.7(b), the situation is saved 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. In Fig. 4.7(c), we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.

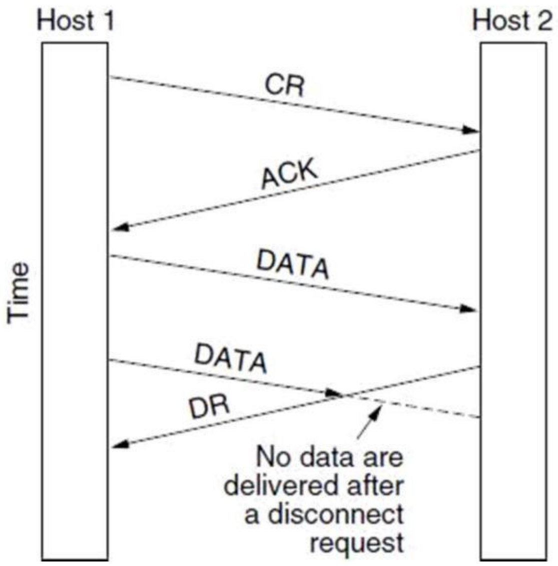


Figure 4.6: Abrupt disconnection with loss of data

Our last scenario, Fig. 4.7(d), is the same as Fig. 4.7(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost segments. After *N* retries, the sender just gives up and releases the connection. Meanwhile, the receiver times out and also exits.

Error control and Flow control:

Error control is ensuring that the data is delivered with the desired level of reliability, usually that all of the data is delivered without any errors. Flow control is keeping a fast transmitter from overrunning a slow receiver.

MULTIPLEXING:

Multiplexing, or sharing several conversations over 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 segment comes in, some way is needed to tell which process to give it to. This situation, called **multiplexing**, is shown in Fig. 4.8(a). In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.
Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a host has multiple network paths that it can use. If a user needs more bandwidth or more reliability than one of the network paths can provide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in Fig. 4.8(b).

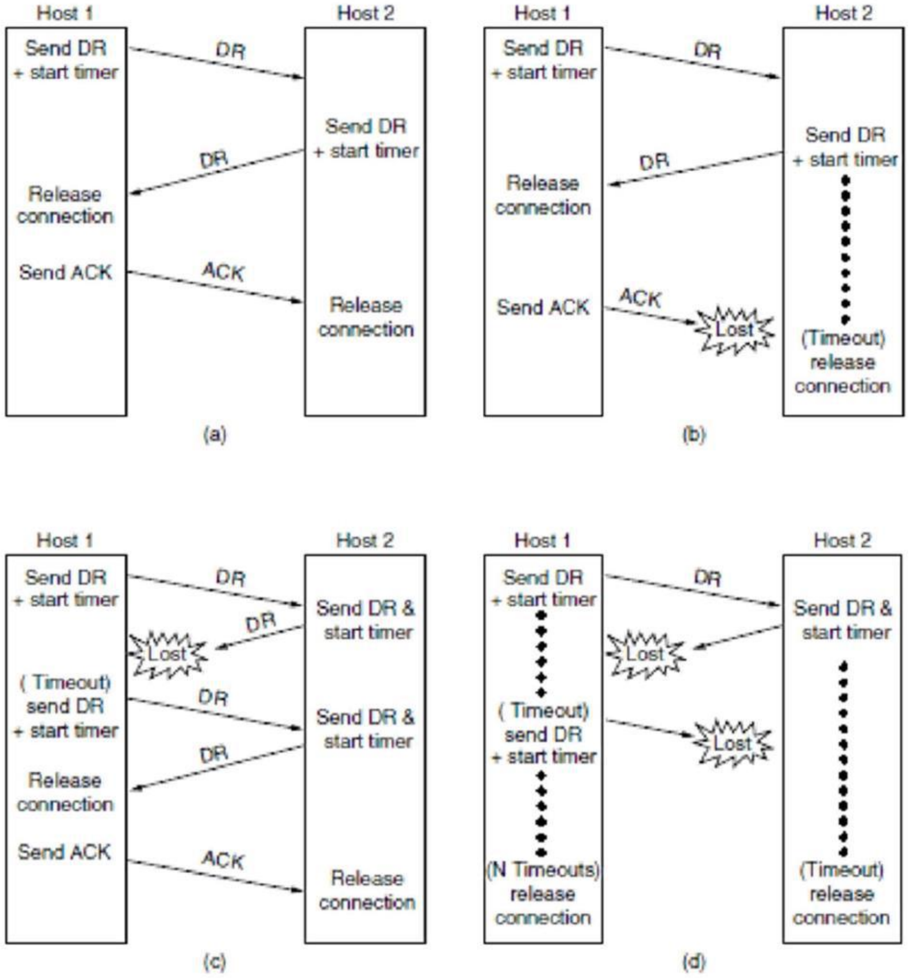


Figure 4.7: Four protocol scenarios for releasing a connection. (a) normal case of three-way handshake. (b) final ACK lost. (c) response lost. (d) response lost and subsequent DRs lost.

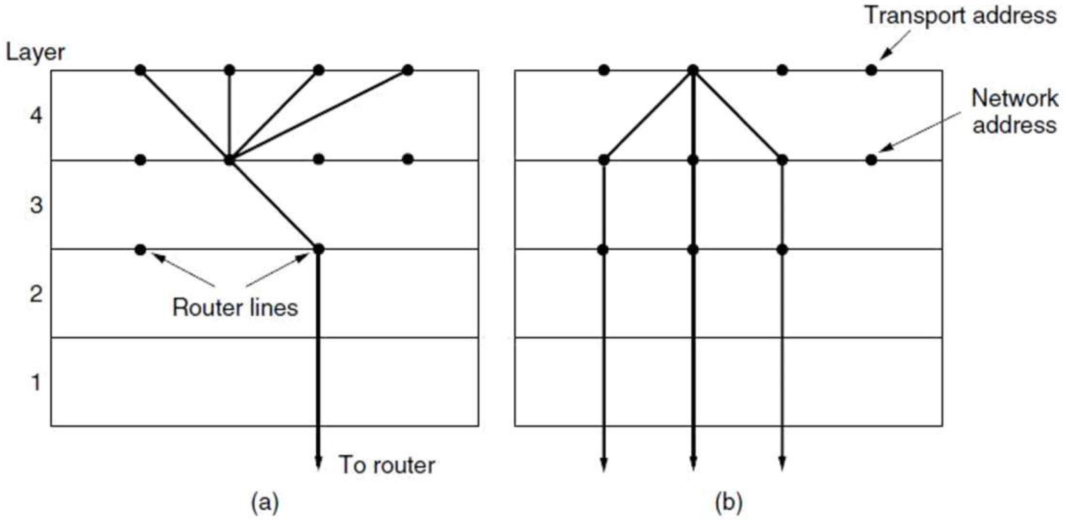


Figure 4.8: (A) Multiplexing (B) Inverse Multiplexing

CRASH RECOVERY:

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads), recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The transport entities expect lost segments all the time and know how to cope with them by using retransmissions. 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 quickly reboot.

CONGESTION CONTROL:

If the transport entities on many machines send too many packets into the network too quickly, the network will become congested, with performance degraded as packets are delayed and lost.

Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers. Congestion occurs at routers, so it is detected at the network layer. However, congestion is ultimately caused by traffic sent into the network by the transport layer. The only effective way to control congestion is for the transport protocols to send packets into the network more slowly.

DESIRABLE BANDWIDTH ALLOCATION:

Before we describe how to regulate traffic, we must understand what we are trying to achieve by running a congestion control algorithm. That is, we must specify the state in which a good congestion control algorithm will operate the network. The goal is more than to simply avoid congestion. It is to find a good allocation of bandwidth to the transport entities that are using the network. A good allocation will deliver good performance because it uses all the available bandwidth but avoids congestion, it will be fair across competing transport entities, and it will quickly track changes in traffic demands.

Efficiency and Power:

An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, five transport entities should get 20 Mbps each. They should usually get less than 20 Mbps for good performance.

Max-Min Fairness:

In the preceding discussion, we did not talk about how to divide bandwidth between different transport senders. This sounds like a simple question to answer— give all the senders an equal fraction of the bandwidth—but it involves several considerations. Perhaps the first consideration is to ask what this problem has to do with congestion control. A second consideration is what a fair portion means for flows in a network. It is simple enough if *N* flows use a single link, in which case they can all have 1/*N* of the bandwidth (although efficiency will dictate that they use slightly less if the traffic is bursty).

But what happens if the flows have different, but overlapping, network paths? For example, one flow may cross three links, and the other flows may cross one link. The three-link flow consumes more network resources. It might be fairer in some sense to give it less bandwidth than the one-link flows. The form of fairness that is often desired for network usage is **max-min fairness**. An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger.

Convergence:

A final criterion is that the congestion control algorithm converge quickly to a fair and efficient allocation of bandwidth. The discussion of the desirable operating point above assumes a static network environment. However, connections are always coming and going in a network, and the bandwidth needed by a given connection will vary over time too. Because of the variation in demand, the ideal operating point for the network varies over time.

A good congestion control algorithm should rapidly converge to the ideal operating point, and it should track that point as it changes over time. If the convergence is too slow, the algorithm will never be close to the changing operating point. If the algorithm is not stable, it may fail to converge to the right point in some cases, or even oscillate around the right point.

Regulating the sending rate:

Now it is time to regulate the sending rates to obtain a desirable bandwidth allocation. The sending rate may be limited by two factors. The first is flow control, in the case that there is insufficient buffering at the receiver.

The second is congestion, in the case that there is insufficient capacity in the network.

In Fig. 4.9, we see this problem illustrated hydraulically. In Fig. 4.9(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation. As long as the sender does not send more water than the bucket can contain, no water will be lost.

In Fig. 4.9(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network. If too much water comes in too fast, it will back up and some will be lost (in this case, by overflowing the funnel).

The way that a transport protocol should regulate the sending rate depends on the form of the feedback returned by the network. Different network layers may return different kinds of feedback. The feedback may be explicit or implicit, and it may be precise or imprecise.

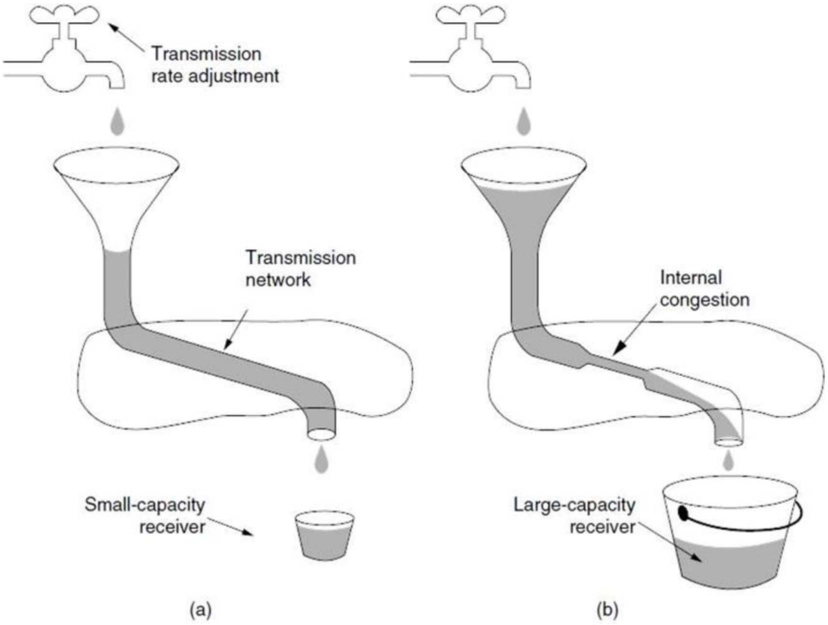


Figure 4.9: (a) a fast network feeding a low-capacity receiver. (b) a slow network feeding a high-capacity receiver.

Wireless issues:

Transport protocols such as TCP that implement congestion control should be independent of the underlying network and link layer technologies. That is a good theory, but in practice there are issues with wireless networks. The main issue is that packet loss is often used as a congestion signal, including by TCP.

Wireless networks lose packets all the time due to transmission errors. To function well, the only packet losses that the congestion control algorithm should observe are losses due to insufficient bandwidth, not losses due to transmission errors. One solution to this problem is to mask the wireless losses by using retransmissions over the wireless link.

THE INTERNET TRANSPORT PROTOCOLS:

UDP:

The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. The protocols complement each other.The connectionless protocol is UDP. It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.The connection-oriented protocol is TCP. It does almost everything. It makes connections and adds reliability with retransmissions, along with flow control and congestion control, all on behalf of the applications that use it.

INTRODUCTION TO UDP:

The Internet protocol suite supports a connectionless transport protocol called **UDP** (**User Datagram Protocol**).UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection. UDP is described in RFC 768. UDP transmits **segments** consisting of an 8-byte header followed by the payload. The header is shown in Fig. 4.10. The two **ports** serve to identify the endpoints within the source and destination machines.

When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when the BIND primitive or something similar is used.

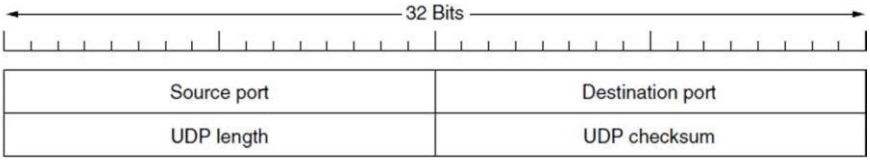


Figure 4.10: the UDP header

Think of ports as mailboxes that applications can rent to receive packets. In fact, the main value of UDP over just using raw IP is the addition of the source and destination ports.
Without the port fields, the transport layer would not know what to do with each incoming packet. With them, it delivers the embedded segment to the correct application.
The source port is primarily needed when a reply must be sent back to the source. By copying the *Source port* field from the incoming segment into the *Destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.

The *UDP length* field includes the 8-byte header and the data. The minimum length is 8 bytes, to cover the header. The maximum length is 65,515 bytes, which is lower than the largest number that will fit in 16 bits because of the size limit on IP packets.
An optional *Checksum* is also provided for extra reliability. It checksums the header, the data, and a conceptual IP pseudoheader. When performing this computation, the *Checksum* field is set to zero and the data field is padded out with an additional zero byte if its length is an odd number. The checksum algorithm is simply to add up all the 16-bit words in one’s complement and to take the one’s complement of the sum.

Remote procedure call:

In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language. The idea behind RPC is to make a remote procedure call look as much as possible like a local one.
In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub**, that represents the server procedure in the client’s address space.

Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local. The actual steps in making an RPC are shown in Fig. 4.12.

- □ Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
- □ Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.
- □ Step 3 is the operating system sending the message from the client machine to the server machine.

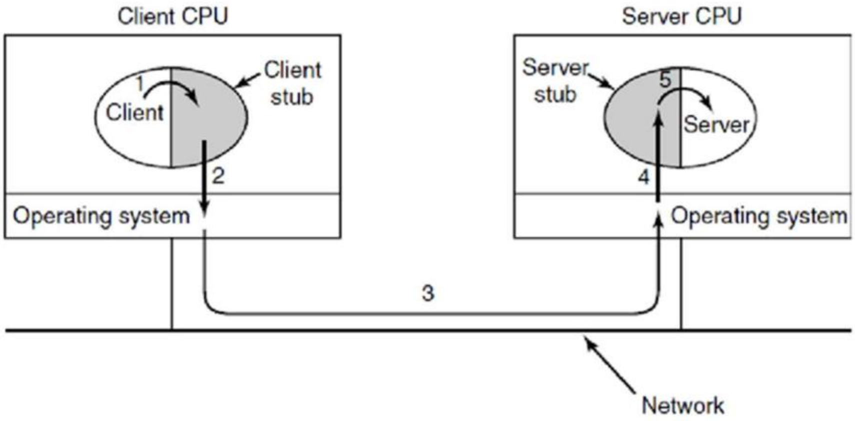


Figure 4.12: Steps in making a remote procedure call, the stubs are shaded

□ Step 4 is the operating system passing the incoming packet to the server stub.

□ Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

The reply traces the same path in the other direction.The key item to note here is that the client procedure, written by the user, just makes a normal (i.e., local) procedure call to the client stub, which has the same name as the server procedure. Since the client procedure and client stub are in the same address space, the parameters are passed in the usual way. Similarly, the server procedure is called by a procedure in its address space with the parameters it expects. To the server procedure, nothing is unusual.

Real-Time Transport Protocols

Client-server RPC is one area in which UDP is widely used. Another one is for real-time multimedia applications. In particular, as Internet radio, Internet telephony, music-on-demand, videoconferencing, video-on-demand, and other multimedia applications became more commonplace, people have discovered that each application was reinventing more or less the same real-time transport protocol. Thus was **RTP** (**Real-time Transport Protocol**) born.

It is described in RFC 3550 and is now in widespread use for multimedia applications. There are two aspects of real-time transport . The first is the RTP protocol for transporting audio and video data in packets. The second is the processing that takes place, mostly at the receiver, to play out the audio and video at the right time.

RTP—The Real-Time Transport Protocol:

The basic function of RTP is to multiplex several real-time data streams onto a single stream of UDP packets. The UDP stream can be sent to a single destination (unicasting) or to multiple destinations (multicasting).

Because RTP just uses normal UDP, its packets are not treated specially by the routers unless some normal IP quality-of-service features are enabled. In particular, there are no special guarantees about delivery, and packets may be lost, delayed, corrupted, etc. The RTP format contains several features to help receivers work with multimedia information. The RTP header is illustrated in Fig. 4.13. It consists of three 32-bit words and potentially some extensions.

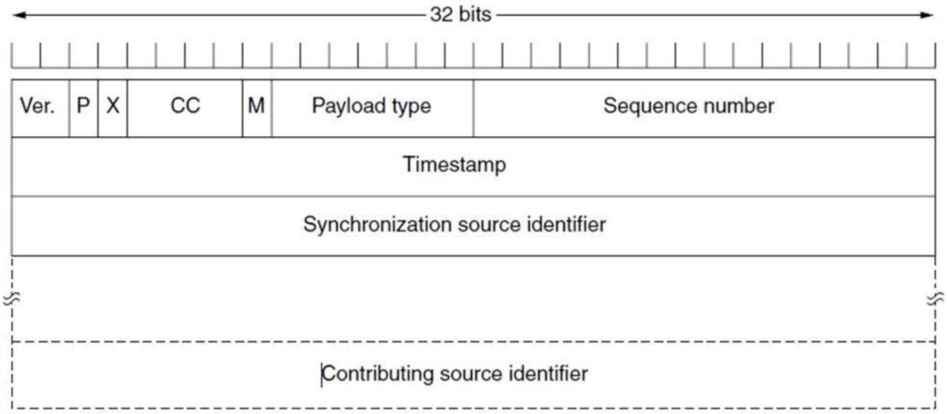

32 bits
Ver.
P
X
CC
M
Payload type
Sequence number
Timestamp
Synchronization source identifier
Contributing source identifier

Figure 4.13: The RTP header

The first word contains the Version field, which is already at 2. The *P* bit indicates that the packet has been padded to a multiple of 4 bytes. The X bit indicates that an extension header is present. The CC field tells how many contributing sources are present, from 0 to 15. The M bit is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or something else that the application understands. The Payload type field tells which encoding algorithm has been used (e.g., uncompress ed 8-bit audio, MP3, etc.). The Sequence number is just a counter that is incremented on each RTP packet sent. It is used to detect lost packets. The Timestamp is produced by the stream's source to note when the first sample in the packet was made. The Synchronization source identifier tells which stream the packet belongs to. It is the method used to multiplex and demultiplex multiple data streams onto a single stream of UDP packets.

Finally, the Contributing source identifiers, if any, are used when mixers are present.

RTCP—The Real-time Transport Control Protocol

RTP has a little sister protocol (little sibling protocol?) called **RTCP** (**Realtime Transport Control Protocol**). It is defined along with RTP in RFC 3550 and handles feedback, synchronization, and the user interface. It does not transport any media samples.

THE INTERNET TRANSPORT PROTOCOLS:

TCP

UDP is a simple protocol and it has some very important uses, such as clientserver interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed. UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet.

Introduction to TCP:

TCP (**Transmission Control Protocol**) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures. TCP was formally defined in RFC 793 in September 1981.

As time went on, many improvements have been made, and various errors and inconsistencies have been fixed. To give you a sense of the extent of TCP, the important RFCs are now RFC 793 plus: clarifications and bug fixes in RFC 1122; extensions for high-performance in RFC 1323. Selective acknowledgements in RFC 2018; congestion control in RFC 2581; repurposing of header fields for quality of service in RFC 2873; improved retransmission timers in RFC 2988; and explicit congestion notification in RFC 3168. The IP layer gives no guarantee that datagrams will be delivered properly, nor any indication of how fa st datagrams may be sent.

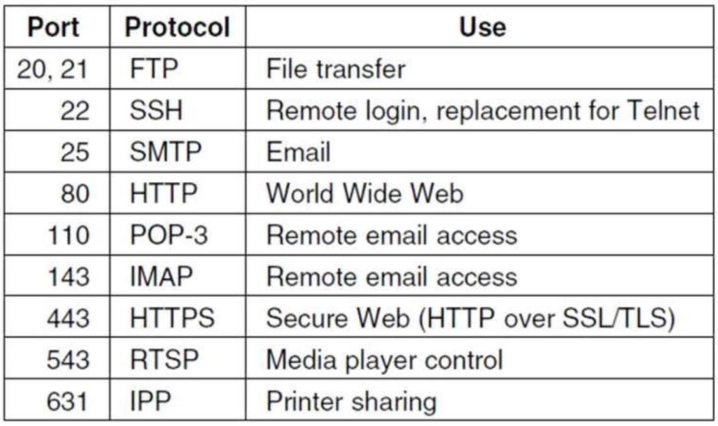
It is up to TCP to send datagrams fast enough to make use of the capacity but not cause congestion, and to time out and retransmit any datagrams that are not delivered. Datagrams that do arrive may well do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence.

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 (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**. A port is the TCP name for a TSAP.

For TCP service to be obtained, a connection must be explicitly established between a socket on one machine and a socket on another machine. A socket may be used for multiple connections at the same time. In other words, two or more connections may terminate at the same socket.

Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users (e.g., root in UNIX systems). They are called **well-known ports**.
For example, any process wishing to remotely retrieve mail from a host can connect to the destination host's port 143 to contact its IMAP daemon. The list of well-known ports is given at *www.iana.org*. Over 700 have been assigned. A few of the better-known ones are listed in Fig. 4.14.



Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Figure 4.14: Some assigned ports

All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points. TCP does not support multicasting or broadcasting. A TCP connection is a byte stream, not a message stream. Message boundaries are not preserved end to end.

The TCP Protocol:

A key feature of TCP, and one that dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number. When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host blasting away at full speed took over 1 week to cycle through the sequence numbers.

The sending and receiving TCP entities exchange data in the form of segments. A **TCP segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be.

It can accumulate data from several writes into one segment or can split data from one write over multiple segments. Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515- byte IP payload. Second, each link has an **MTU** (**Maximum Transfer Unit**).

Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet. However, it is still possible for IP packets carrying TCP segments to be fragmented when passing over a network path for which some link has a small MTU. If this happens, it degrades performance and causes other problems. Instead, modern TCP implementations perform **path MTU discovery** by using the technique outlined in RFC 1191. This technique uses ICMP error messages to find the smallest MTU for any link on the path. TCP then adjusts the segment size downwards to avoid fragmentation.

The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size. When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, and otherwise without) bearing an acknowledgement number equal to the next sequence number it expects to receive and the remaining window size. If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header:

Figure 4.15 shows the layout of a TCP segment. Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options.

After the options, if any, up to 65,535 − 20 − 20 = 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header.

Segments without any data are legal and are commonly used for acknowledgements and control message

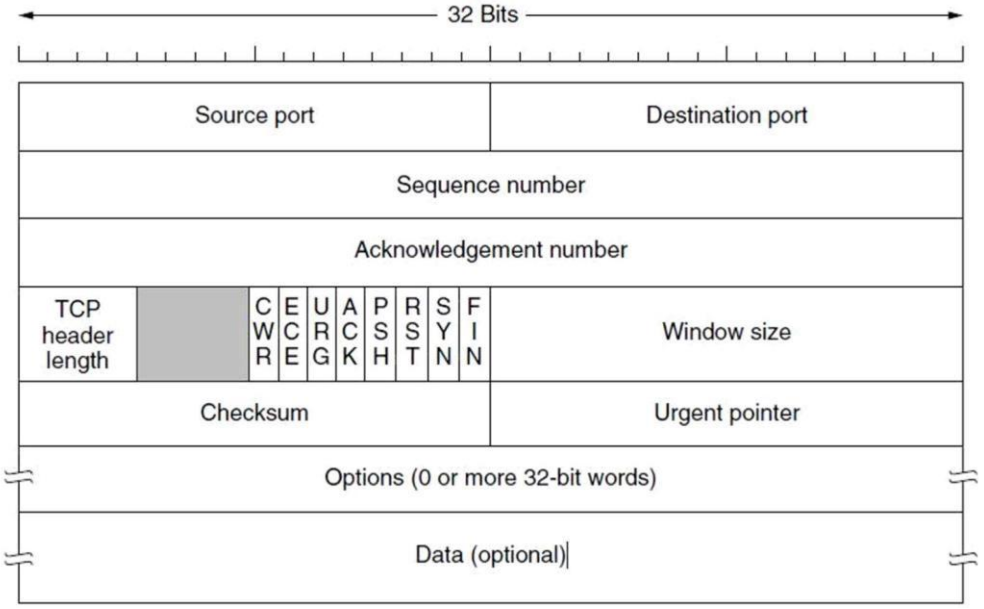


Figure 4.15: The TCP Header

The *Source port* and *Destination port* fields identify the local end points of the connection. The source and destination end points together identify the connection. This connection identifier is called a **5 tuple** because it consists of five pieces of information: the protocol (TCP), source IP and source port, and destination IP and destination port.

The *Sequence number* and *Acknowledgement number* fields perform their usual functions.
The *Sequence number* and *Acknowledgement number* fields perform their usual functions.
The *TCP header length* tells how many 32-bit words are contained in the TCP header. This information is needed because the *Options* field is of variable length, so the header is, too.
Now come eight 1-bit flags. *CWR* and *ECE* are used to signal congestion when ECN (Explicit Congestion Notification) is used. *CWR* is set to signal *Congestion Window Reduced* from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the *ECN-Echo*.

URG is set to 1 if the *Urgent pointer* is in use. The *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
The *ACK* bit is set to 1 to indicate that the *Acknowledgement number* is valid. This is the case for nearly all packets. If *ACK* is 0, the segment does not contain an acknowledgement, so the *Acknowledgement number* field is ignored.

The *PSH* bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received (which it might otherwise do for efficiency).
The *RST* bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason.
The *SYN* bit is used to establish connections. The *FIN* bit is used to release a connection.
The *Window size* field tells how many bytes may be sent starting at the byte acknowledged.
A *Checksum* is also provided for extra reliability. The *Options* field provides a way to add extra facilities not covered by the regular header.

TCP Connection Establishment:
Connections are established in TCP by means of the three-way handshake. To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives in that order, either specifying a specific source or nobody in particular.

The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password). The CONNECT primitive sends a TCP segment with the *SYN* bit on and *ACK* bit off and waits for a response.

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

TCP Connection Release

Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections. Each simplex connection is released independently of its sibling.

To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit. When the *FIN* is acknowledged, that direction is shut down for new data.

Data may continue to flow indefinitely in the other direction, however. When both directions have been shut down, the connection is released.

TCP Congestion Control:

The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that it is sending into the network.

In the Internet, TCP plays the main role in controlling congestion, as well as the main role in reliable transport. That is why it is such a special protocol.

PERFORMANCE PROBLEMS IN COMPUTER NETWORKS

Some performance problems, such as congestion, are caused by temporary resource overloads. If more traffic suddenly arrives at a router than the router can handle, congestion will build up and performance will suffer.

Performance also degrades when there is a structural resource imbalance. For example, if a gigabit communication line is attached to a low-end PC, the poor host will not be able to process the incoming packets fast enough and some will be lost. These packets will eventually be retransmitted, adding delay, wasting bandwidth, and generally reducing performance.

Overloads can also be synchronously triggered. As an example, if a segment contains a bad parameter , in many cases the receiver will thoughtfully send back an error notification.

Another tuning issue is setting timeouts. When a segment is sent, a timer is set to guard against loss of the segment. If the timeout is set too short, unnecessary retransmissions will occur, clogging the wires. If the timeout is set too long, unnecessary delays will occur after a segment is lost.

NETWORK PERFORMANCE MEASUREMENT:

When a network performs poorly, its users often complain to the folks running it, demanding improvements. To improve the performance, the operators must first determine exactly what is going on. To find out what is really happening, the operators must make measurements.

Measurements can be made in different ways and at many locations (both in the protocol stack and physically). The most basic kind of measurement is to start a timer when beginning some activity and see how long that activity takes.

Other measurements are made with counters that record how often some event has happened (e.g., number of lost segments). Measuring network performance and parameters has many potential pitfalls. We list a few of them here. Any systematic attempt to measure network performance should be careful to avoid these.

1) Make Sure That the Sample Size Is Large Enough

Do not measure the time to send one segment, but repeat the measurement, say, one million times and take the average.

2) Make Sure That the Samples Are Representative

Ideally, the whole sequence of one million measurements should be repeated at different times of the day and the week to see the effect of different network conditions on the measured quantity.

3) Caching Can Wreak Havoc with Measurements

Repeating a measurement many times will return an unexpectedly fast answer if the protocols use caching mechanisms.

4) Be Sure That Nothing Unexpected Is Going On during Your Tests

Making measurements at the same time that some user has decided to run a video conference over your network will often give different results than if there is no video conference.

5) Be Careful When Using a Coarse-Grained Clock

Computer clocks function by incrementing some counter at regular intervals.

6) Be Careful about Extrapolating the Results

Suppose that you make measurements with simulated network loads running from 0 (idle) to 0.4 (40% of capacity).

THE APPLICATION LAYER:

INTRODUCTION:

The application layer provides services to the user. Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

Providing Services:

All communication networks that started before the Internet were designed to provide services to network users. Most of these networks, however, were originally designed to provide one specific service. For example, the telephone network was originally designed to provide voice service: to allow people all over the world to talk to each other. This network, however, was later used for some other services, such as facsimile (fax), enabled by users adding some extra hardware at both ends.

The Internet was originally designed for the same purpose: to provide service to users around the world. The layered architecture of the TCP/IP protocol suite, however, makes the Internet more flexible than other communication networks such as postal or telephone networks.

Each layer in the suite was originally made up of one or more protocols, but new protocols can be added or some protocols can be removed or replaced by the Internet authorities. However, if a protocol is added to each layer, it should be designed in such a way that it uses the services provided by one of the protocols at the lower layer.

If a protocol is removed from a layer, care should be taken to change the protocol at the next higher layer that supposedly uses the services of the removed protocol. The application layer, however, is somewhat different from other layers in that it is the highest layer in the suite.

The protocols in this layer do not provide services to any other protocol in the suite; they only receive services from the protocols in the transport layer. This means that protocols can be removed from this layer easily. New protocols can be also added to this layer as long as the new protocols can use the services provided by one of the transport-layer protocols.

Standard and Nonstandard Protocols:

To provide smooth operation of the Internet, the protocols used in the first four layers of the TCP/IP suite need to be standardized and documented.

Standard Application-Layer Protocols:

There are several application-layer protocols that have been standardized and documented by the Internet authority, and we are using them in our daily interaction with the Internet.
Each standard protocol is a pair of computer programs that interact with the user and the transport layer to provide a specific service to the user.

Nonstandard Application-Layer Protocols:

A programmer can create a nonstandard application-layer program if she can write two programs that provide service to the user by interacting with the transport layer.

Application-Layer Paradigms

It should be clear that to use the Internet we need two application programs to interact with each other: one running on a computer somewhere in the world, the other running on another computer somewhere else in the world. The two programs need to send messages to each other through the Internet infrastructure.
However, we have not discussed what the relationship should be between these programs.
Should both application programs be able to request services and provide services, or should the application programs just do one or the other?

Two paradigms have been developed during the lifetime of the Internet to answer this question: the client-server paradigm and the peer-to-peer paradigm.

Traditional Paradigm: Client-Server:

The traditional paradigm is called the **client-server paradigm**. It was the most popular paradigm until a few years ago. In this paradigm, the service provider is an application program, called the server process; it runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.

There are normally some server processes that can provide a specific type of service, but there are many clients that request service from any of these server processes. The server process must be running all the time; the client process is started when the client needs to receive service.

New Paradigm: Peer-to-Peer:

A new paradigm, called the **peer-to-peer paradigm** (often abbreviated *P2P paradigm*) has emerged to respond to the needs of some new applications.
In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect. The responsibility is shared between peers.
A computer connected to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time.

CLIENT-SERVER PROGRAMMING:
In a client-server paradigm, communication at the application layer is between two running application programs called ***processes**:* a *client* and a *server*.
A client is a running program that initializes the communication by sending a request; a server is another application program that waits for a request from a client.
The server handles the request received from a client, prepares a result, and sends the result back to the client. This definition of a server implies that a server must be running when a request from a client arrives, but the client needs to be run only when it is needed.

This means that if we have two computers connected to each other somewhere, we can run a client process on one of them and the server on the other. However, we need to be careful that the server program is started before we start running the client program.

Application Programming Interface:

A client process communicate with a server process with the help of a computer program which is normally written in a computer language with a predefined set of instructions that tells the computer what to do. A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/output access, and so on. If we need a process to be able to communicate with another process, we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection. A set of instructions of this kind is normally referred to as an **application programming interface (API)**.

An interface in programming is a set of instructions between two entities. In this case, one of the entities is the process at the application layer and the other is the *operating system* that encapsulates the first four layers of the TCP/IP protocol sui te.

Several APIs have been designed for communication. One of the most common one is: **socket interface**. The socket interface is a set of instructions that provide communication between the application layer and the operating system, as shown in Figure 5.1.

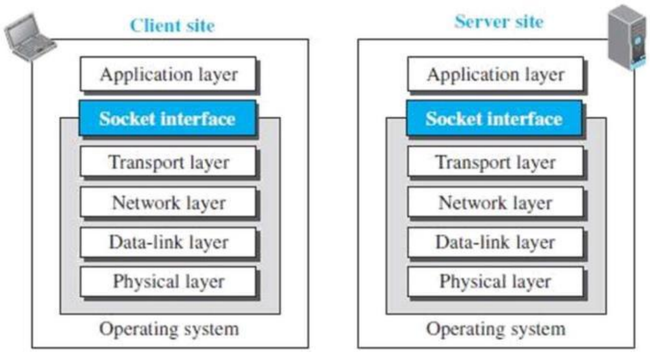


FIGURE 5.1: Position Of The Socket Interface

It is a set of instructions that can be used by a process to communicate with another process. The idea of sockets allows us to use the set of all instructions already designed in a programming language for other sources and sinks.

For example, in most computer languages, like C, C++, or Java, we have several instructions that can read and write data to other sources and sinks such as a keyboard (a source), a monitor (a sink), or a file (source and sink). We can use the same instructions to read from or write to sockets.

Sockets:

Although a socket is supposed to behave like a terminal or a file, it is not a physical entity like them; it is an abstraction. It is an object that is created and used by the application program.

Socket Addresses:

The interaction between a client and a server is two-way communication. In a two-way communication, we need a pair of addresses: local (sender) and remote (receiver). The local address in one direction is the remote address in the other direction and vice versa. Since communication in the client-server paradigm is between two sockets, we need a pair of **socket addresses** for communication: a local socket address and a remote socket address. However, we need to define a socket address in terms of identifiers used in the TCP/IP protocol suite.

A socket address should first define the computer on which a client or a server is running. Socket address should be a combination of an IP address (32 bit) and a port number (16 bit). Since a socket defines the end-point of the communication, we can say that a socket is identified by a pair of socket addresses, a local and a remote.

Finding Socket Addresses: How can a client or a server find a pair of socket addresses for communication? The situation is different for each site.

Server Site: The server needs a local (server) and a remote (client) socket address for communication.

Local Socket Address The local (server) socket address is provided by the operating system. The operating system knows the IP address of the computer on which the server process is running. The port number of a server process, however, needs to be assigned.

If the server process is a standard one defined by the Internet authority, a port number is already assigned to it. For example, the assigned port number for a Hypertext Transfer Protocol (HTTP) is the integer 80, which cannot be used by any other process.

Remote Socket Address The remote socket address for a server is the socket address of the client that makes the connection. Since the server can serve many clients, it does not know beforehand the remote socket address for communication.

The server can find this socket address when a client tries to connect to the server. The client socket address, which is contained in the request packet sent to the server, becomes the remote socket address that is used for responding to the client.

Client Site: The client also needs a local (client) and a remote (server) socket address for communication.

Local Socket Address The local (client) socket address is also provided by the operating system. The operating system knows the IP address of the computer on which the client is running. The port number, however, is a 16-bit temporary integer that is assigned to a client process each time the process needs to start the communication. The port number, however, needs to be assigned from a set of integers defined by the Internet authority and called the ephemeral (temporary) port numbers. The operating system, however, needs to guarantee that the new port number is not used by any other running client process.

Remote Socket Address Finding the remote (server) socket address for a client, however, needs more work. When a client process starts, it should know the socket address of the server it wants to connect to.

Using Services of the Transport Layer:

A pair of processes provide services to the users of the Internet, human or programs. A pair of processes, however, need to use the services provided by the transport layer for communication because there is no physical communication at the application layer.

WORLD WIDE WEB AND HTTP:

World Wide Web: The idea of the Web was first proposed by Tim Berners-Lee in 1989. The Web today is a repository of information in which the documents, called ***web pages,*** are distributed all over the world and related documents are linked together.

The popularity and growth of the Web can be related to two terms in the above statement: *distributed* and *linked*. Distribution allows the growth of the Web. Each web server in the world can add a new web page to the repository and announce it to all Internet users without overloading a few servers. Linking allows one web page to refer to another web page stored in another server somewhere else in the world. The linking of web pages was achieved using a concept called ***hypertext,*** which was introduced many years before the advent of the Internet.

The idea was to use a machine that automatically retrieved another document stored in the system when a link to it appeared in the document. The Web implemented this idea electronically to allow the linked document to be retrieved when the link was clicked by the user. Today, the term *hypertext,* coined to mean linked text documents, has been changed to ***hypermedia,*** to show that a web pagecan be a text document, an image, an audio file, or a video file.

Architecture:

The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites.* Each site holds one or more web pages. Each web page, however, can contain some links to other web pages in the same or other sites. In other words, a web page can be simple or composite. A simple web page has no links to other web pages; a composite web page has one or more links to other web pages. Each web page is a file with a name and address.

Web Client (Browser): A variety of vendors offer commercial **browsers** that interpret and display a web page, and all of them use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocols, and interpreters (figure 5.2).

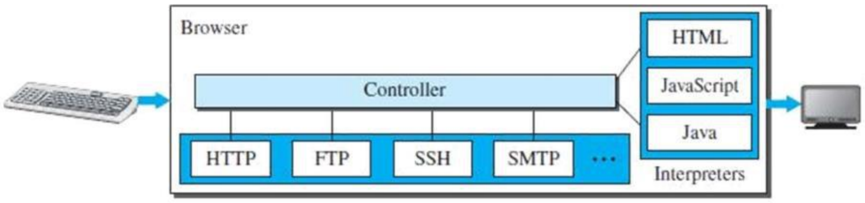


Figure 5.2: Browser

The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described later, such as HTTP or FTP. The interpreter can be HTML, Java, or JavaScript, depending on the type of document. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

Web Server: The web page is stored at the server. Each time a request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than a disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time. Some popular web servers include Apache and Microsoft Internet Information Server.

Uniform Resource Locator (URL):

A web page, as a file, needs to have a unique identifier to distinguish it from other web pages. To define a web page, we need three identifiers: *host*, *port*, and *path*. However, before defining the web page, we need to tell the browser what client server application we want to use, which is called the *protocol*. This means we need four identifiers to define the web page.

The first is the type of vehicle to be used to fetch the web page; the last three make up the combination that defines the destination object (web page).

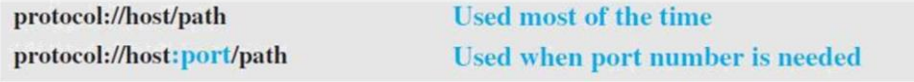
***Protocol*.** The first identifier is the abbreviation for the client-server program that we need in order to access the web page.
Although most of the time the protocol is HTTP (HyperText Transfer Protocol), we can also use other protocols such as FTP (File Transfer Protocol).

***Host*.** The host identifier can be the IP address of the server or the unique name given to the server. IP addresses can be defined in dotted decimal notation.

***Port*.** The port, a 16-bit integer, is normally predefined for the client-server application.

***Path*.** The path identifies the location and the name of the file in the underlying operating system. The format of this identifier normally depends on the operating system.

To combine these four pieces together, the **uniform resource locator (URL)** has been designed; it uses three different separators between the four pieces as shown below:



protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

Web Documents:
The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static Documents:
Static documents are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. In other words, the contents of the file are determined when the file is created, not when it is used. Static documents are prepared using one of several languages: *HyperText Markup Language* (HTML), *Extensible Markup Language* (XML), *Extensible Style Language* (XSL), and *Extensible Hypertext Markup Language* (XHTML).

Dynamic Documents:
A **dynamic document** is created by a web server whenever a browser requests the document. When a request arrives, the web server runs an application program or a script that creates the dynamic document. The server returns the result of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server.

Active Documents:
For many applications, we need a program or a script to be run at the client site. These are called ***active documents*.** For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user.

HyperText Transfer Protocol (HTTP):
The **HyperText Transfer Protocol (HTTP)** is used to define how the client-server programs can be written to retrieve web pages from the Web. An HTTP client sends a request; an HTTP server returns a response. The server uses the port number 80; the client uses a temporary port number. HTTP uses the services of TCP, which, as discussed before, is a connection-oriented and reliable protocol.

Nonpersistent versus Persistent Connections:
If the web pages, objects to be retrieved, are located on different servers, we do not have any other choice than to create a new TCP connection for retrieving each object. However, if some of the objects are located on the same server, we have two choices: to retrieve each object using a new TCP connection or to make a TCP connection and retrieve them all. The first method is referred to as a *nonpersistent connection*, the second as a *persistent connection*.

Nonpersistent Connections
In a **nonpersistent connection**, one TCP connection is made for each request/response.
The following lists the steps in this strategy:

1. The client opens a TCP connection and sends a request.

2. The server sends the response and closes the connection.

3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

Persistent Connections

HTTP version 1.1 specifies a **persistent connection** by default. In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response. However, there are some occasions when the sender does not know the length of the data.

This is the case when a document is created dynamically or actively. In these cases, the server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached. Time and resources are saved using persistent connections.
Only one set of buffers and variables needs to be set for the connection at each site. The round trip time for connection establishment and connection termination is saved.

Message Formats:

The HTTP protocol defines the format of the request and response messages. Each message is made of four sections. The first section in the request message is called the *request line;* the first section in the response message is called the *status line*. The other three sections have the same names in the request and response messages. However, the similarities between these sections are only in the names; they may have different contents. We discuss each message type separately.

Request Message:

There are three fields in this line separated by one space and terminated by two characters (carriage return and line feed). The fields are called *method, URL*, and *version*.
The method field defines the request types. Several methods are defined like GET, PUT, HEAD, POST, TRACE, DELETE, etc. The URL defines the address and name of the corresponding web page. The version field gives the version of the protocol; the most current version of HTTP is 1.1.

Response Message:

A response message consists of a status line, header lines, a blank line, and sometimes a body. The first line in a response message is called the *status line*. There are three fields in this line separated by spaces and terminated by a carriage return and line feed.
The first field defines the version of HTTP protocol, currently 1.1. The status code field defines the status of the request. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site.

The status phrase explains the status code in text form. After the status line, we can have zero or more *response header* lines. Each header line sends additional information from the server to the client.

Web Caching: Proxy Servers:

HTTP supports **proxy servers**. A proxy server is a computer that keeps copies of responses to recent requests. The HTTP client sends a request to the proxy server. The proxy server checks its cache. If the response is not stored in the cache, the proxy server sends the request to the corresponding server. Incoming responses are sent to the proxy server and stored for future requests from other clients.

The proxy server reduces the load on the original server, decreases traffic, and improves latency. However, to use the proxy server, the client must be configured to access the proxy instead of the target server.

HTTP Security:

HTTP per se does not provide security. HTTP can be run over the Secure Socket Layer (SSL). In this case, HTTP is referred to as HTTPS. HTTPS provides confidentiality, client and server authentication, and data integrity.

FTP:

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first.

Although we can transfer files using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats. Figure 5.3 shows the basic model of FTP. The client has three components: the user interface, the client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process.

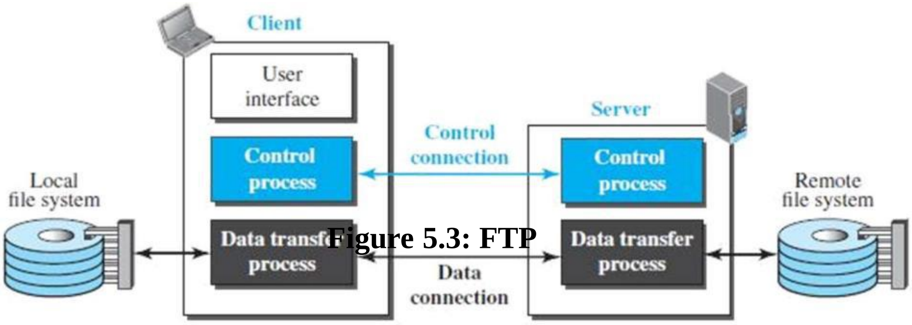


Figure 5.3: FTP

The control connection is made between the control processes. The data connection is made between the data transfer processes. Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.

Two Connections

The two connections in FTP have different lifetimes. The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transfer activity.

FTP uses two well-known TCP ports: port 21 is used for the control connection, and port 20 is used for the data connection.

Control Connection:
During this control connection, commands are sent from the client to the server and responses are sent from the server to the client. Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. Some of the most common commands are shown in table below:

Command	*Argument(s)*	*Description*
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	Port identifier	Client chooses a port
PWD		Display name of current directory
QUIT		Log out of the system

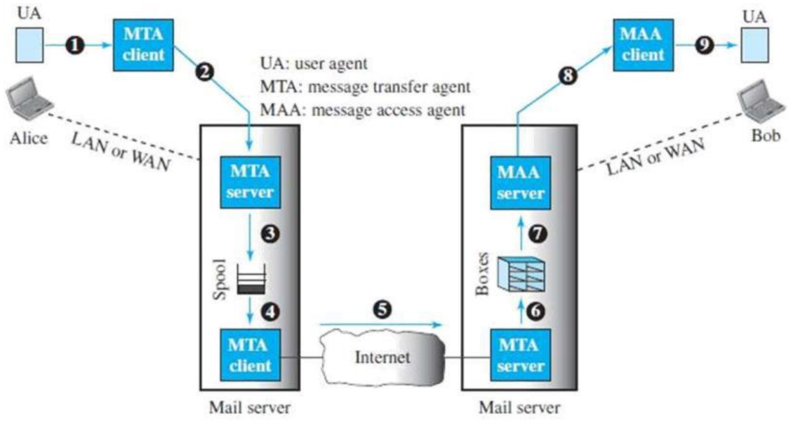
Every FTP command generates at least one response. A response has two parts: a three-digit number followed by text. The numeric part defines the code; the text part defines needed parameters or further explanations. The first digit defines the status of the command. The second digit defines the area in which the status applies. The third digit provides additional information.

Code	Description	Code	Description
125	Data Connection Open	250	Request file action OK
150	File Status OK	331	User name OK; password is needed
200	Command OK	425	Cannot open data connection

ELECTRONIC MAIL: Electronic mail (or e-mail) allows users to exchange messages. The nature of this application, however, is different from other applications discussed so far. In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client. When the request arrives, the server provides the service. There is a request and there is a response.

In the case of electronic mail, the situation is different. First, e-mail is considered a one-way transaction. When Alice sends an email to Bob, she may expect a response, but this is not a mandate. Bob may or may not respond. If he does respond, it is another one -way transaction.
Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it.
This means that the idea of client/server programming should be implemented in another way: using some intermediate computers (servers). The users run only client programs when they want and the intermediate servers apply the client/server paradigm

Architecture:
To explain the architecture of e-mail, we give a common scenario as shown in Figure 5.4.



In the common scenario, the sender and the receiver of the e-mail, Alice and Bob respectively, are connected via a LAN or a WAN to two mail servers. The administrator has created one mailbox for each user where the received messages are stored. A *mailbox* is part of a server hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. The administrator has also created a queue (spool) to store messages waiting to be sent.
A simple e-mail from Alice to Bob takes nine different steps. Alice and Bob use three different *agents*: a **user agent (UA),** a **message transfer agent (MTA),** and a **message access agent (MAA).** When Alice needs to send a message to Bob, she runs a UA program to prepare the message and send it to her mail server.

The mail server at her site uses a queue (spool) to store messages waiting to be sent. The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an MTA. Here two message transfer agents are needed: one client and one server.
Like most client-server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection. The client, on the other hand, can be triggered by the system when there is a message in the queue to be sent.

The user agent at the Bob site allows Bob to read the received message. Bob later uses an MAA client to retrieve the message from an MAA server running on the second server.
User Agent: The first component of an electronic mail system is the **user agent (UA).** It provides service to the user to make the process of sending and receiving a message easier. A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers.
Message Transfer Agent: SMTP: Based on the common scenario, we can say that the e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task. It is important that we distinguish these three when we are dealing with e-mail. The formal protocol that defines the MTA client and server in the Internet is called ***Simple Mail Transfer Protocol (SMTP).*** SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. SMTP simply defines how commands and responses must be sent back and forth.

Message Access Agent: POP and IMAP: The first and second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from the client to the server.On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. The third stage uses a message access agent.
Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).

POP3: Post Office Protocol, version 3 (POP3) is simple but limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server. Mail access starts with the client when the user needs to download its e -mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. POP3 has two modes: the *delete* mode and the *keep* mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval.

IMAP4: Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4).** IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex. POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. In addition, POP3 does not allow the user to partially check the contents of the mail before downloading. IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.

TELNET:

A server program can provide a specific service to its corresponding client program. For example, the FTP server is designed to let the FTP client store or retrieve files on the server site. However, it is impossible to have a client/server pair for each type of service we need; the number of servers soon becomes intractable which is not scalable. Another solution is to have a specific client/server program for a set of common scenarios, but to have some generic client/server programs that allow a user on the client site to log into the computer at the server site and use the services available there.

For example, if a student needs to use the Java compiler program at her university lab, there is no need for a Java compiler client and a Java compiler server. The student can use a client logging program to log into the university server and use the compiler program at the university. We refer to these generic client/server pairs as ***remote logging*** applications. One of the original remote logging protocols is **TELNET,** which is an abbreviation for *TErminaL NETwork.* Although TELNET requires a logging name and password, it is vulnerable to hacking because it sends all data including the password in plaintext (not encrypted).

A hacker can eavesdrop and obtain the logging name and password. Because of this security issue, the use of TELNET has diminished in favor of another protocol, Secure Shell (SSH).

Although TELNET is almost replaced by SSH, we briefly discuss TELNET here for two reasons:

1. The simple plaintext architecture of TELNET allows us to explain the issues and challenges related to the concept of remote logging, which is also used in SSH when it serves as a remote logging protocol.

2. Network administrators often use TELNET for diagnostic and debugging purposes.

Local versus Remote Logging:

When a user logs into a local system, it is called *local logging*. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.

The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

However, when a user wants to access an application program or utility located on a remote machine, she performs *remote logging.* Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them.

The characters are sent to the TELNET client, which transforms the characters into a universal character set called *Network Virtual Terminal* (NVT) characters and delivers them to the local TCP/IP stack. The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server; it is designed to receive characters from a terminal driver.

The solution is to add a piece of software called a *pseudoterminal driver*, which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

NVT uses two sets of characters, one for data and one for control. Both are 8 -bit bytes. For data, NVT normally uses what is called *NVT ASCII*. This is an 8-bit character set in which the seven lowest order bits are the same as US ASCII and the highest order bit is 0.

To send control characters between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest order bit is set to 1.

Options: TELNET lets the client and server negotiate options before or during the use of the service.

User Interface:

The operating system (UNIX, for example) defines an interface with user-friendly commands. An example of such a set of commands can be found in Table beow:

Command Name	**Meaning**
open	Connect to a remote computer
close	Close the connections
display	Show the operating parameters
mode	Change to line or character mode
Quit	Exit TELNET
Send	Send special characters

SECURE SHELL (SSH): Although **Secure Shell (SSH)** is a secure application program that can be used today for several purposes such as remote logging and file transfer, it was originally designed to replace TELNET. There are two versions of SSH: SSH-1 and SSH-2, which are totally incompatible. The first version, SSH-1, is now deprecated because of security flaws in it. In this section, we discuss only SSH-2.

Components: SSH is an application-layer protocol with three components.

SSH Transport-Layer Protocol (SSH-TRANS):
Since TCP is not a secured transport-layer protocol, SSH first uses a protocol that creates a secured channel on top of the TCP. This new layer is an independent protocol referred to as SSH-TRANS. When the procedure implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure connection. Then they exchange several security parameters to establish a secure channel on top of the TCP. The services provided by this protocol are:

1. Privacy or confidentiality of the message exchanged.

2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder.

3. Server authentication, which means that the client is now sure that the server is the one that it claims to be.

4. Compression of the messages, which improves the efficiency of the system and makes attack more difficult.

SSH Authentication Protocol (SSH-AUTH):
After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another procedure that can authenticate the client for the server. The client authentication process in SSH is very similar to what is done in Secure Socket Layer (SSL). This layer defines a number of authentication tools similar to the ones used in SSL. Authentication starts with the client, which sends a request message to the server. The request includes the user name, server name, the method of authentication, and the required data. The server responds with either a success message, which confirms that the client is authenticated, or a failed message, which means that the process needs to be repeated with a new request message.

SSH Connection Protocol (SSH-CONN):
After the secured channel is established and both server and client are authenticated for each other, SSH can call a piece of software that implements the third protocol, SSHCONN. One of the services provided by the SSH-CONN protocol is multiplexing. SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it. Each channel can be used for a different purpose, such as remote logging, file transfer, and so on.

Applications: Although SSH is often thought of as a replacement for TELNET, SSH is, in fact, a general-purpose protocol that provides a secure connection between a client and server.

SSH for Remote Logging: Several free and commercial applications use SSH for remote logging. Among them, we can mention PuTTy, by Simon Tatham, which is a client SSH program that can be used for remote logging. Another application program is Tectia, which can be used on several platforms.

SSH for File Transfer: One of the application programs that is built on top of SSH for file transfer is the *Secure File Transfer Program* (*sftp*). The *sftp* application program uses one of the channels provided by the SSH to transfer files. Another common application is called *Secure Copy* (*scp*). This application uses the same format as the UNIX copy command, *cp,* to copy files.

DOMAIN NAME SYSTEM (DNS): Since the Internet is so huge today, a central directory system cannot hold all the mapping. In addition, if the central computer fails, the whole communication network will collapse. A better solution is to distribute the information among many computers in the world. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the **Domain Name System (DNS)**.

Figure 5.5 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address. A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server n ame, such as *afilesource.com.*

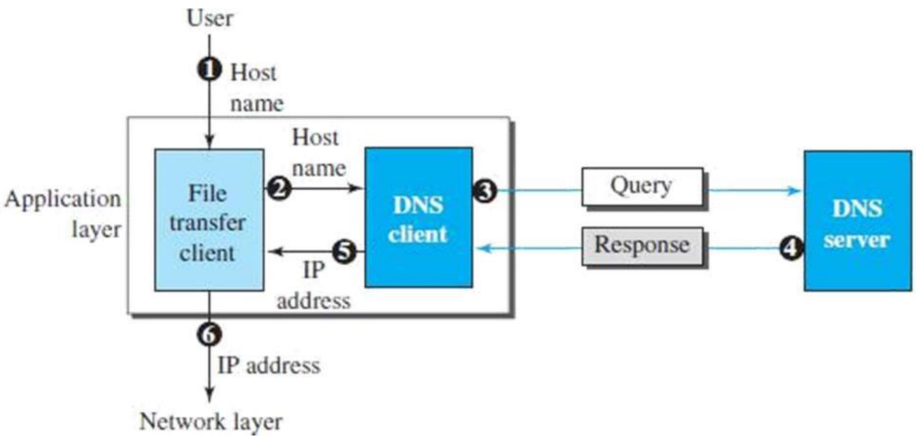


Figure 5.5: Purpose of DNS

Name Space:
A **name space** that maps each address to a unique name can be organized in two ways: flat or hierarchical. In a *flat name space,* a name is assigned to an address.

A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication. In a *hierarchical name space,* each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized.

A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility for the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the same, the whole address is different.

Domain Name Space:
To have a hierarchical name space, a **domain name space** was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127 (see Figure 5.6).

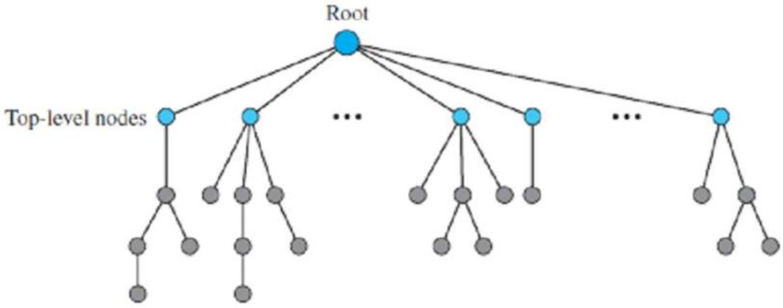


Figure 5.6: Domain name space

Label:

Each node in the tree has a **label**, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name:

Each node in the tree has a domain name. A full **domain name** is a sequence of labels separated by dots (**.**). The domain names are always read from the node up to the root.

The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Figure 5.7 shows some domain names.

Domain:

A **domain** is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree. Figure 5.8 shows some domains. Note that a domain may itself be divided into domains.

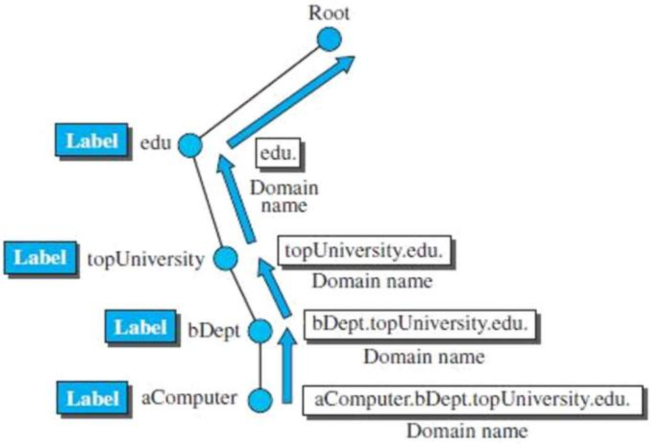


Figure 5.7: Domain names and labels

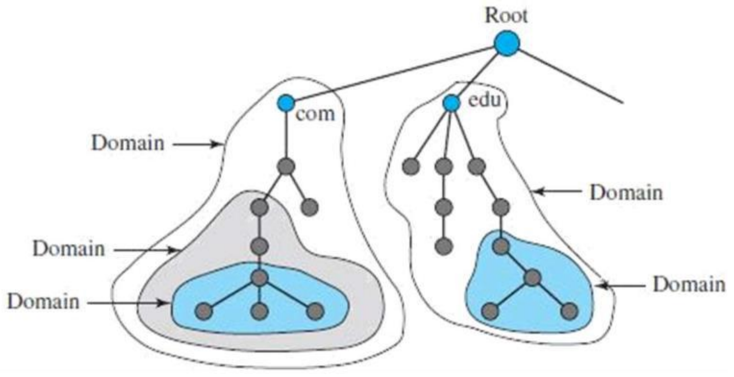


Figure 5.8: Domains

SNMP:

Several network management standards have been devised during the last few decades. The most important one is **Simple Network Management Protocol (SNMP),** used by the Internet. SNMP is a framework for managing devices in an internet using the TCP/IP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet. SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers (see Figure 5.9).

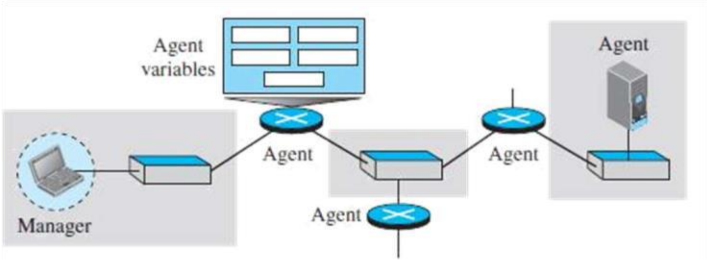


Figure 5.9: SNMP concept

SNMP is an application-level protocol in which a few manager stations control a set of agents. The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks. In other words, SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology. It can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers.

Managers and Agents: A management station, called a *manager*, is a host that runs the SNMP client program. A managed station, called an *agent,* is a router (or a host) that runs the SNMP server program. Management is achieved through simple interaction between a manager and an agent. The agent keeps performance information in a database. The manager has access to the values in the database.

For example, a router can store in appropriate variables the number of packets received and forwarded. The manager can fetch and compare the values of these two variables to see if the router is congested or not.

The manager can also make the router perform certain actions. For example, a router periodically checks the value of a reboot counter to see when it should reboot itself. It reboots itself, for example, if the value of the counter is 0. The manager can use this feature to reboot the agent remotely at any time. It simply sends a packet to force a 0 value in the counter. Agents can also contribute to the management process. The server program running on the agent can check the environment and, if it notices something unusual, it can send a warning message (called a *Trap*) to the manager. In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by warning the manager of an unusual situation.

Management Components: To do management tasks, SNMP uses two other protocols: **Structure of Management Information (SMI)** and
Management Information Base (MIB).

Role of SNMP: SNMP has some very specific roles in network management. It defines the format of the packet to be sent from a manager to an agent and vice versa. It also interprets the result and creates statistics (often with the help of other management software).

Role of SMI: To use SNMP, we need rules for naming objects. This is particularly important because the objects in SNMP form a hierarchical structure. Part of a name can be inherited from the parent. We also need rules to define the types of objects.

Role of MIB: MIB creates a set of objects defined for each entity in a manner similar to that of a database (mostly metadata in a database, names & types without values).

UNIT-5
Security Concepts

Most initial computer applications had no or at best, very little security.The need for

security:

1. ***Protecting the functionality of the organization:***

   The decision maker in organizations must set policy and operates their organization in compliance with the complex, shifting legislation, efficientand capable applications.

2. ***Enabling the safe operation of applications:***

   The organization is under immense pressure to acquire and operates integrated, efficient and capable applications. The modern organization needs to create an environment that safeguards application using the organizations IT systems, particularly those application that serves as important elements of the infrastructure of the organization.

3. ***Protecting the data that the organization collect and use:***

   Data in the organization can be in two forms are either in rest or in motion, the motion of data signifies that data is currently used or processed by the system. The values of the data motivated the attackersto steal or corrupts the data. This is essential for the integrity and the values of the organization‟s data. Information security ensures the protection of both data in motion as well as data in rest.

4. ***Safeguarding technology assets in organizations:***

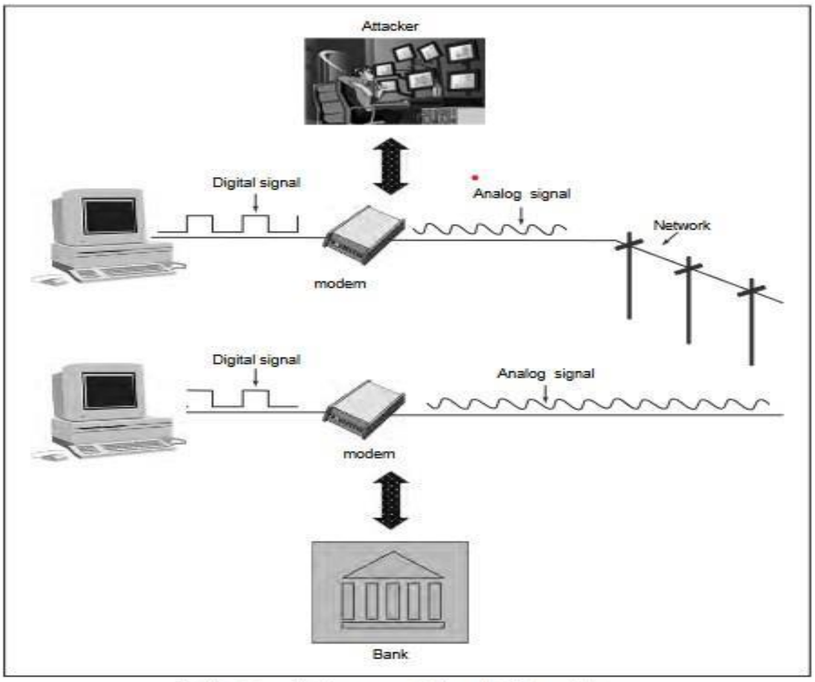
   The organization must add intrastate services based on the size and scope of the organization. Organizational growth could lead to the need for public key infrastructure, PKI an integrated system of the software, encryption methodologies. The information security mechanism used by large organizations is complex in comparison to a small organization. The small organization generally prefers symmetric key encryption of data.

SECURITY APPROACHES

1. **Trusted Systems:**

A trusted system is a computer system that can be trusted to a specified extent to enforce a specified security policy.

Trusted systems were initially of primary interest to the military. However, these days, the concept has spanned across various areas, most prominently in the banking and financial community, but the concept never caught on. Trusted systems often use the term reference monitor.



⊢ Fig. 1.3 *Attacks can now be launched from a distance*

It is mainly responsible for all the decisions related to access controls.Naturally, following are the expectations from the reference monitor:

(a) It should be tamperproof

(b) It should always be invoked

(c) It should be small enough so that it can be independently tested

2. *Security Models*

An organization can take several approaches to implement its security model.Let us summarize theseapproaches.

- **No security**In this simplest case, the approach could be a decision toimplement no security at all.
- **Security through obscurity**In this model, a system is secure simply because nobody knows about its existence and contents. This approach cannot work for too long, as there are many ways an attackercan come to know about it.
- **Host security**In this scheme, the security for each host is enforced individually. This is a very safe approach, but the trouble is that it cannotscale well. The complexity and diversity of modern sites/organizations makes the task even harder.
- **Network security** Host security is tough to achieve as organizations growand become more diverse. In this technique, the focus is to controlnetwork access to various hosts and their services, rather than individual host security. This is a very efficient and scalable model

3. *Security Management Practices*

Good security management practices always talk of a security policy being inplace. Putting a securitypolicy in place is actually quite tough.

A good security policy generally takes care offour key aspects, as follows:

- Affordability Cost and effort in security implementation.
- Functionality Mechanism of providing security.
- Cultural issues Whether the policy gels well with people‟s expectations,working style and beliefs.
- Legality Whether the policy meets the legal requirements.

Once a security policy is in place, the following points should be ensured.

(a) Explanation of the policy to all concerned.

(b) Outline everybody‟s responsibilities.

(c) Use simple language in all communications.

(d) Establishment of accountability.

(e) Provision for exceptions and periodic reviews.

PRINCIPLES OF SECURITY

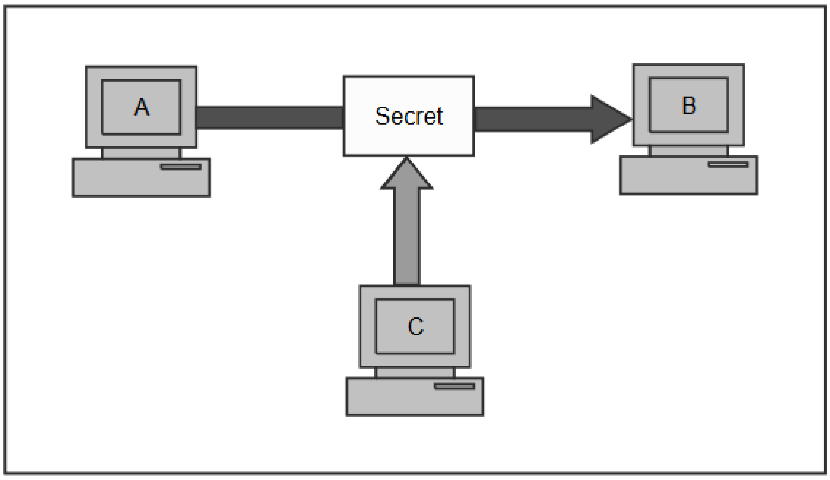
There are six principles

1. Confidentiality.
2. Authentication.
3. Integrity.
4. Non-repudiation.
5. Access control
6. Availability

.

1. *Confidentiality*

The principle of confidentiality specifies that only the sender and the intended recipient(s) should be able to access the contents of a message. Confidentiality gets compromised if an unauthorized person is able to access a message. Example of compromising the confidentiality of a message is shown in Fig. Here, the user of computer A sends a message to user of computer B.



Loss of confidentiality

Another user C gets access to this message, which is not desired andtherefore, defeats the purpose of confidentiality. Example of this could be a confidential email messagesent by A to B, which is accessed by C without the permission or knowledge of A and B. This type ofattack is called as interception.

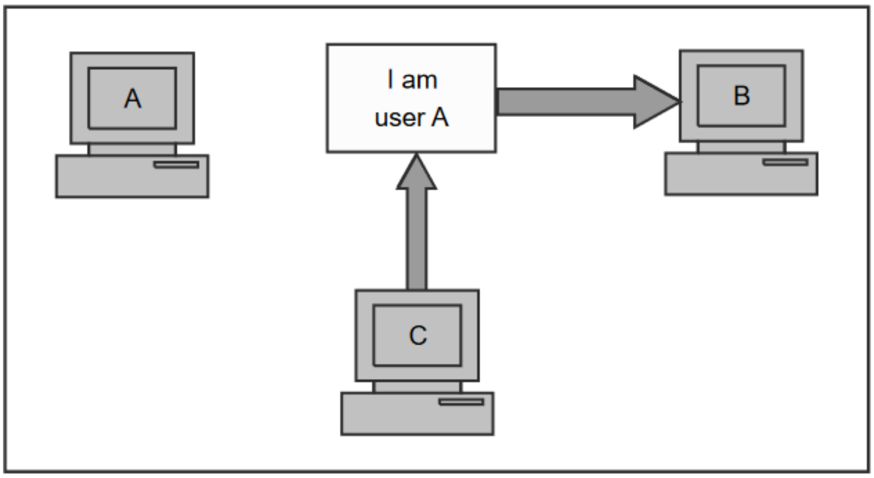
Interception causes loss of message confidentiality.

2. *Authentication*

Authentication mechanisms help establish proof of identities. The authentication process ensures thatthe origin of a electronic message or document is correctly identified.

suppose that user Csends an electronic document over the Internet to user B. However, the trouble is that user C had posedas user A when she sent this document to user B.

A real life example of this could be the case of a user C, posing asuser A, sending a funds transfer request (from A‟s account to C‟s account) to bank B. The bank mighthappily transfer the funds from A‟s account to C‟s account – after all, it would think that user A hasrequested for the funds transfer! This concept is shown in Fig.



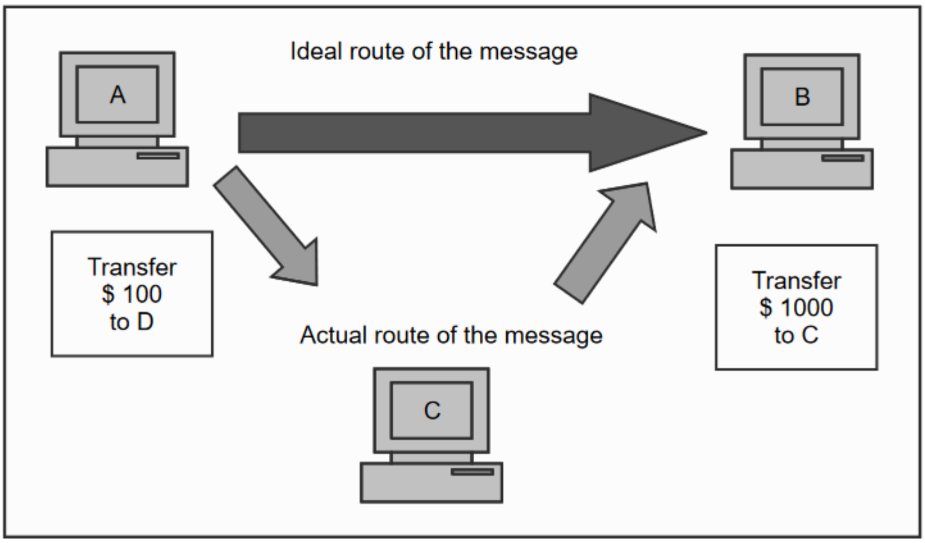
3. Integrity

When the contents of a message are changed after the sender sends it, but before it reaches the intendedrecipient, we say that the integrity of the message is lost. For example, suppose you write a check forRs. 100 to pay for the goods bought from the US. However, when you see your next account statement, youare startled to see that the check resulted in a payment of Rs. 1000. This is the case for loss of messageintegrity. Conceptually, this is shownin Fig.

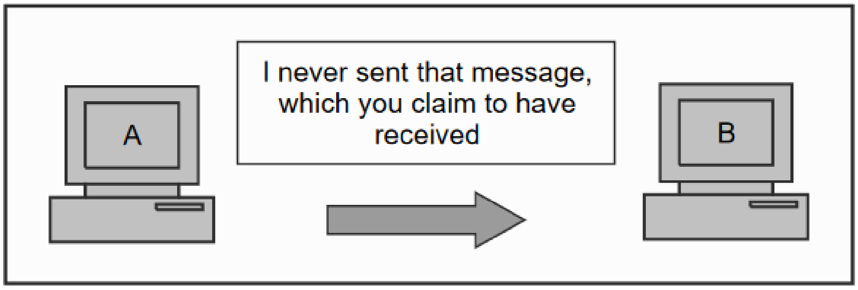
Loss of integrity

4. Non-repudiation

There are situations where a user sends a message and later on refuses that she had sent that message. Forinstance, user A could send a funds transfer request to bank B over the Internet. After the bank performsthe funds transfer as per A‟s instructions, A could claim that she never sent the funds transfer instructionto the bank! Thus, A repudiates or denies, her funds



transfer instruction. The principle of non-repudiation defeats such possibilitiesof denying something, having done it. This is shown in Fig.



Establishing the non-repudiation

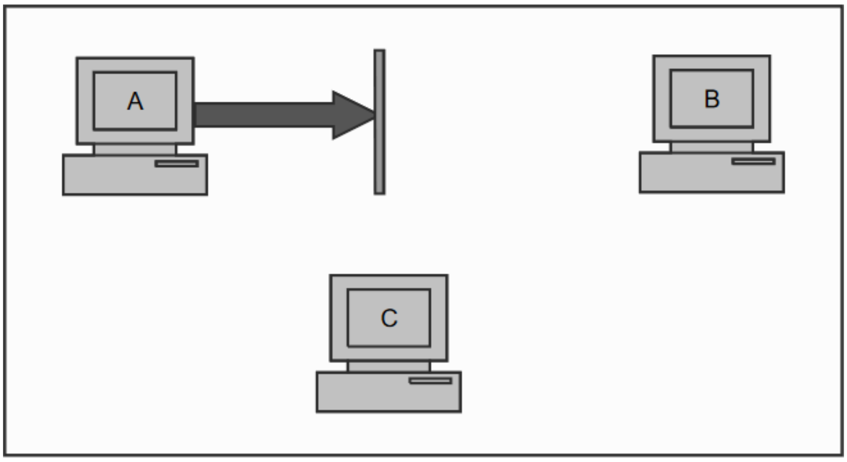
5. *Access Control*

The principle of access control determines who should be able to access what. For instance, we should be able to specify that user A can view the records in a database, but cannot update them. However, user B might be allowed to make updates as well. An access control mechanism can be set up to ensure this. Access control is broadly related to two areas: role management and rule management. Role management concentrates on the user side (which user can do what), whereas rule management focuses on the resources side (which resource is accessible and under what circumstances).

6. *Availability*

The principle of availability states that resources (i.e. information) should be available to authorized parties at all times. For example, due to the intentional actions of an unauthorized user C, an authorized user A may not be able to contact a server computer B, as shown in Fig.

Attack on availability

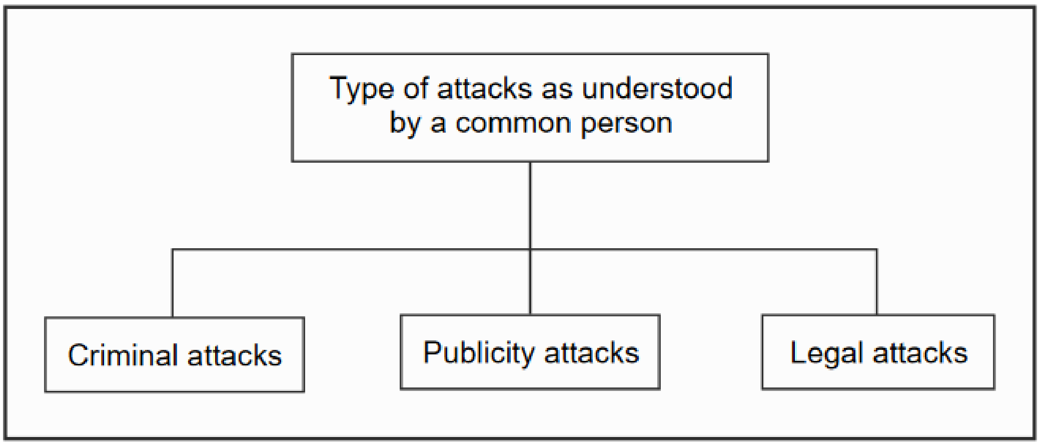


TYPES OF SECURITY ATTACKS

We shall classify attacks with respect to two views: the common **person's view** and a **technologist's view.**

1. *General Attacks:*

A General ViewFrom a common person"s point of view, we can classify attacks into three categories, as shown inFig.

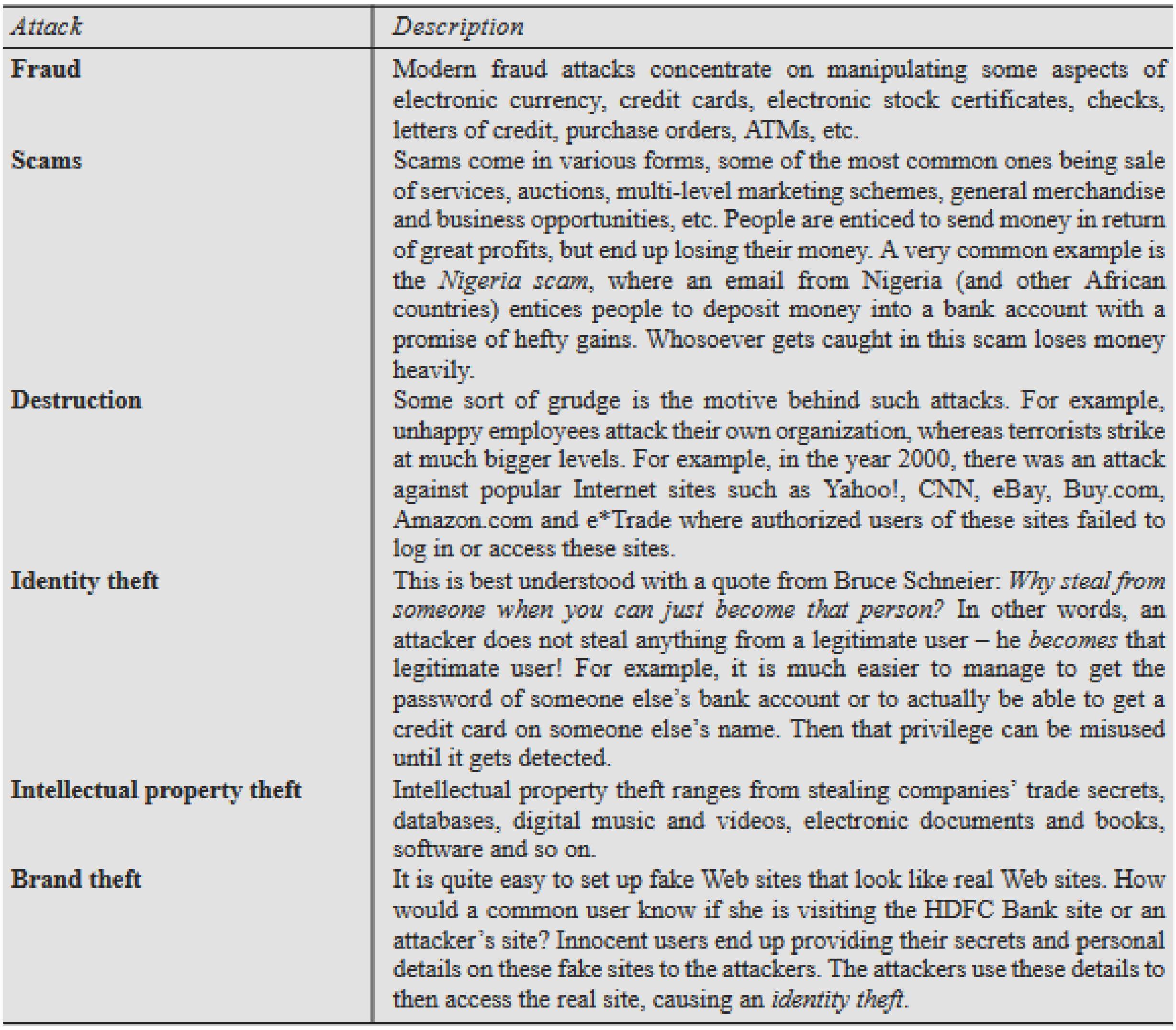


Classification of attacks in general terms

Criminal Attacks Criminal attacks are the simplest to understand. Here, thesole aim of the attackers is to maximize financial gain by attacking computer systems. The **following table gives** some of the criminal attacks.

Publicity Attacks Publicity attacks occur because the attackers want to seetheir names appear on television news channels and newspapers. Historysuggests that these types of attackers are usually not hardcore criminals. Theyare people such as students in universities or employees in large organizations, who seek publicity by adopting a novel approach of attacking computer systems.

Legal Attacks This form of attack is quite novel and unique. Here, the attacker tries to make the judge or the jury doubtful about the security of a computer system. This works as follows. The attacker attacks the computer system and the attacked party (say a bank or an organization) manages to take the attackerto the court.



Attack	*Description*
Fraud	Modern fraud attacks concentrate on manipulating some aspects of electronic currency, credit cards, electronic stock certificates, checks, letters of credit, purchase orders, ATMs, etc.
Scams	Scams come in various forms, some of the most common ones being sale of services, auctions, multi-level marketing schemes, general merchandise and business opportunities, etc. People are enticed to send money in return of great profits, but end up losing their money. A very common example is the *Nigeria scam*, where an email from Nigeria (and other African countries) entices people to deposit money into a bank account with a promise of hefty gains. Whosoever gets caught in this scam loses money heavily.
Destruction	Some sort of grudge is the motive behind such attacks. For example, unhappy employees attack their own organization, whereas terrorists strike at much bigger levels. For example, in the year 2000, there was an attack against popular Internet sites such as Yahoo!, CNN, eBay, Buy.com, Amazon.com and e*Trade where authorized users of these sites failed to log in or access these sites.
Identity theft	This is best understood with a quote from Bruce Schneier: *Why steal from someone when you can just become that person?* In other words, an attacker does not steal anything from a legitimate user – he *becomes* that legitimate user! For example, it is much easier to manage to get the password of someone else's bank account or to actually be able to get a credit card on someone else's name. Then that privilege can be misused until it gets detected.
Intellectual property theft	Intellectual property theft ranges from stealing companies' trade secrets, databases, digital music and videos, electronic documents and books, software and so on.
Brand theft	It is quite easy to set up fake Web sites that look like real Web sites. How would a common user know if she is visiting the HDFC Bank site or an attacker's site? Innocent users end up providing their secrets and personal details on these fake sites to the attackers. The attackers use these details to then access the real site, causing an *identity theft*.

2. *ATTACKS: A TECHNICAL VIEW*

From the technical point of view, we can classify the types of attacks oncomputers and network systems into two categories for better understanding:

(a) Theoretical concepts behind these attacks.

(b)Practical approaches used by the attackers.

(a) *Theoretical Concepts*

These attacks are generally classified into four categories.

- **Interception** –It means that an unauthorized party has gained access to aresource. The party can be a person, program or computer-based system. Examples of interception are copying of data or programs and listening to network traffic.
- **Fabrication** –This involves creation of illegal objects on a computer system. For example, the attacker may add fake records to a database.
  - **Modification** –For example the attacker may modify the values in a database.

- **Interruption** – Here, the resource becomes unavailable, lost or unusable. Examples of interruption are causing problems to a hardware device, erasing program, data or operating system components.

These attacks are further grouped into twotypes:

- Passive attacks.
- Active attacks.

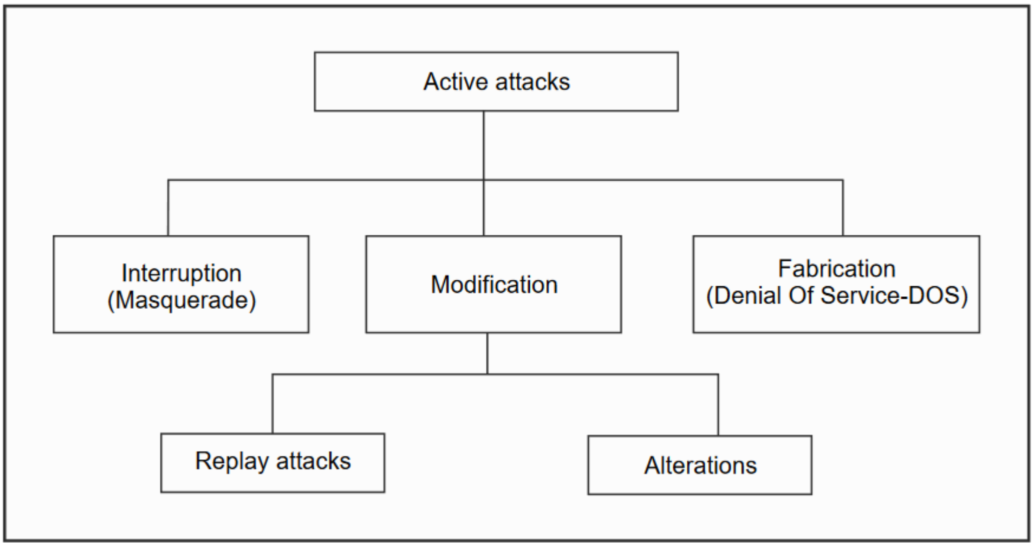
Passive attacks: Passive attacks are those, wherein the attacker indulges in eavesdropping or monitoring of data transmission. In other words, the attacker aims to obtain information that is in transit. The term passive indicates that the attacker does not attempt to perform any modifications to the data.

Passive attacks do not involve any modifications to the contents of an original message.

Active attacks Unlike passive attacks, the active attacks are based onmodification of the original message in some manner or the creation of a falsemessage. These attacks cannot be prevented easily. However, they can bedetected with some effort and attempts can be made to recover from them.These attacks can be in the form of interruption, modification and fabrication. *In active attacks, the contents of the original message are modified in someway.*

Active attacks

Masquerade is caused when an unauthorized entity pretends to be anotherentity.


Active attacks
Interruption (Masquerade)
Modification
Fabrication (Denial Of Service-DOS)
Replay attacks
Alterations

Replay attack, a user captures a sequence of events or some data units and re-sends them.

Alteration of messages involves some change to the original message. Forinstance, suppose user A sends an electronic message Transfer $1000 to D‟s account to bank B. User C might capture this and change it to Transfer $10000 to C"s account.

Denial Of Service (DOS) attacks make an attempt to prevent legitimate users from accessing some services, which they are eligible for. For instance, an unauthorized user might send too many login requests to a server using random user ids one after the other in quick succession, so as to flood the network and deny other legitimate users from using the network facilities.

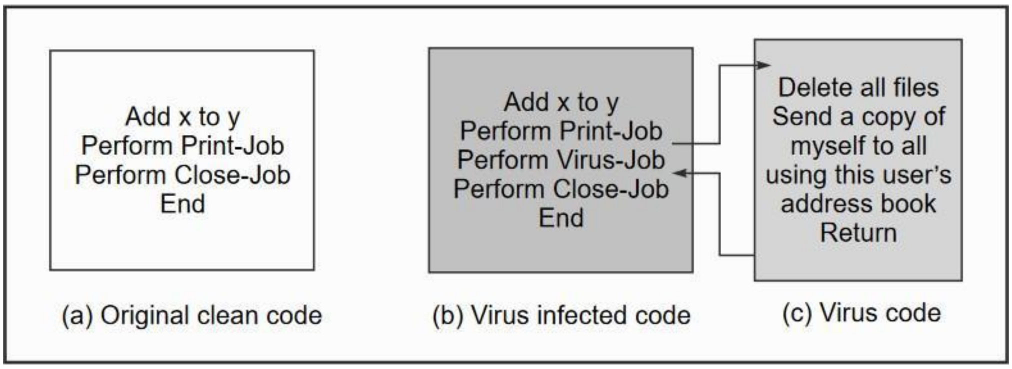
3. *PROGRAMS THAT ATTACK*

Let us now discuss a few programs that attack computer systems to cause some damage or to create confusion.

Virus One can launch an application-level attack or a network level attack usinga virus. In simple terms, a ***virus*** *is a piece of program code that attaches itself to legitimate program code and runs when the legitimate program runs. It can then infect other programs in that computer or programs that are in other computers but on the same network.*

Virus

A virus is a computer program that attaches itself to another legitimate program and causes damage to the computer system or to the network.



During its lifetime, a virus goes through four phases:

(a) **Dormant phase:** Here, the virus is idle. It gets activated based on certain action or event (e.g. the user typing a certain key or certain date or time is reached, etc). This is an optional phase.

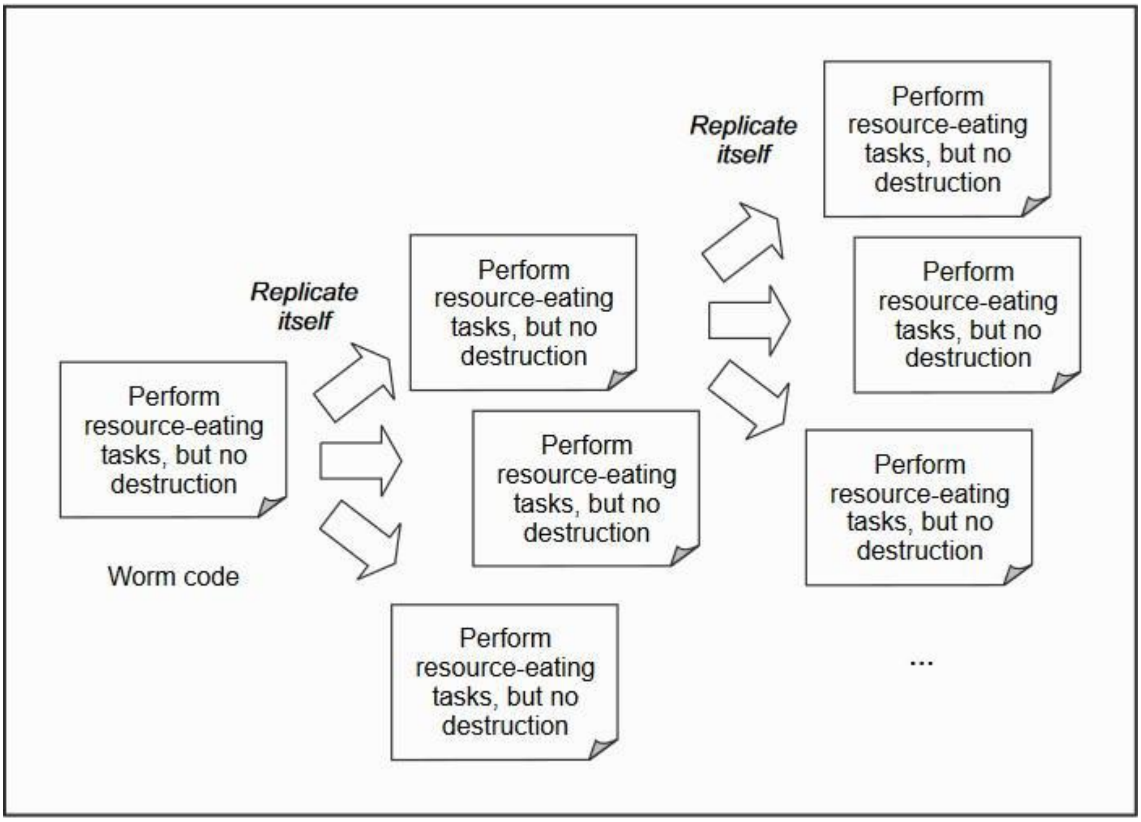
(b) **Propagation phase:** In this phase, a virus copies itself and each copy starts creating more copies of self, thus propagating the virus.

(c) **Triggering phase:** A dormant virus moves into this phase when the action/event for which it was waiting is initiated.

(d) **Execution phase:** This is the actual work of the virus, which could be harmless (display some message on the screen) or destructive (delete a file on the disk).

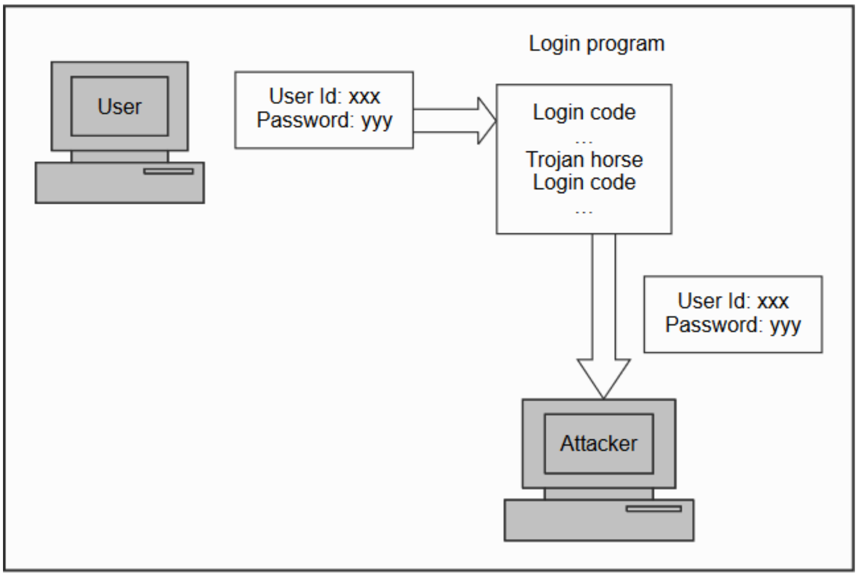
Worm Similar in concept to a virus, a worm is actually different in implementation. A virus modifies a program (i.e. it attaches itself to the program under attack). A worm, however, does not modify a program. Instead, itreplicates itself again and again.

Worm


Perform resource-eating tasks, but no destruction
Worm code
Replicate itself
Perform resource-eating tasks, but no destruction
Perform resource-eating tasks, but no destruction
Perform resource-eating tasks, but no destruction
Replicate itself
Perform resource-eating tasks, but no destruction
Perform resource-eating tasks, but no destruction
Perform resource-eating tasks, but no destruction
...

Trojan Horse A Trojan horse is a hidden piece of code, like a virus. However, the purpose of aTrojan horse is different. Whereas the main purpose of a virus is to make some sort of modifications to the target computer or network, a Trojan horse attempts to reveal confidential information to an attacker.

A Trojan horse allows an attacker to obtain some confidential information abouta computer or a network.



Trojan horse

4. *Specific Attacks*

There are two specific attacks.

1. Sniffing
2. Spoofing

On the Internet, computers exchange messages with each other in the form of small blocks of data, called as packets. A packet, like a postal envelope contains the actual data to be sent and the addressing information. Attackers target these packets, as they travel from the source computer to the destination computer over the Internet.

These attacks take two main forms:

(a) Packet sniffing

(b) Packet spoofing

(a) Packet sniffing: Packet sniffing is a passive attack on an on-going conversation. An attacker need not hijack a conversation, but instead, cansimply observe (i.e. sniff) packets as they pass by.

Clearly, to prevent an attacker from sniffing packets, the information that is passing needs to be protected in some ways.

This can be done at two levels:

(i) The data that is traveling can be encoded in some ways

(ii) The transmission link itself can be encoded.

To read a packet, the attacker somehow needs to access it in the first place.

(B) Packet spoofing: In this technique, an attacker sends packets with a false source address. When this happens, the receiver (i.e. the party who receives these packets containing false address) would inadvertently send replies back tothis forged address (called as spoofed address) and not to the attacker.

This can lead to three possible cases:

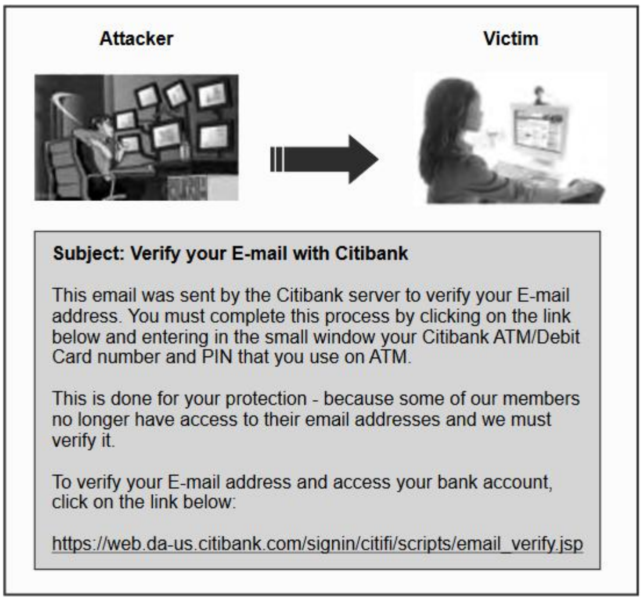
(i) The attacker can intercept the reply – If the attacker is between the destination and the forged source, the attacker can see the reply and use that information for hijacking attacks.

(ii) The attacker need not see the reply – If the attacker‟s intention was a Denial Of Service (DOS) attack, the attacker need not bother about the reply.

(iii) The attacker does not want the reply – The attacker could simply be angry with the host,so it may put that host‟s address as the forged source address and send the packet to the destination.

Phishing has become a big problem in recent times. The attacker‟s module works as follows

- The attacker decides to create her own Web site, which looks veryidentical to a real Web site. For example, the attacker can clone Citibank‟s Web site. The cloning is so clever that human eye will not be able to distinguish between the real (Citibank‟s) and fake (attacker‟s) sites now
- The attacker can use many techniques to attack the bank‟s customers.
- When the customer (i.e. the victim) innocently clicks on the URL specified in the email, she is taken to the attacker‟s site and not the bank‟s original site.



Subject: Verify your E-mail with Citibank

This email was sent by the Citibank server to verify your E-mail address. You must complete this process by clicking on the link below and entering in the small window your Citibank ATM/Debit Card number and PIN that you use on ATM.

This is done for your protection - because some of our members no longer have access to their email addresses and we must verify it.

To verify your E-mail address and access your bank account, click on the link below:

https://web.da-us.citibank.com/signin/citifi/scripts/email_verify.jsp

- There, the customer is prompted to enter confidential information, such as her password or PIN. Since the attacker‟s fake site looks exactlyike the original bank site, the customer provides this information.

Cryptography Concepts and Techniques:

Cryptography is technique of securing information and communications through use of codes so that only those people for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”. One is confidentiality which basically means that we need to be sure that nobody will see our information as it travels across a network. Authentication and access control is also another capability provided by cryptography. Some other capabilities provided by cryptography are non- repudiation and integrity.

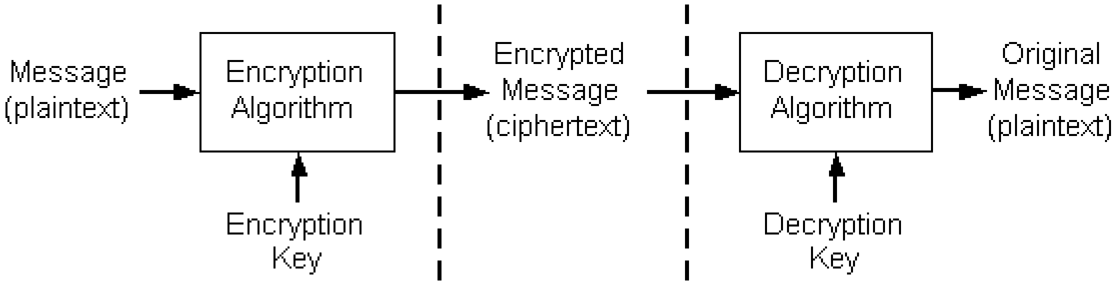
In Cryptography the techniques which are used to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. Thesealgorithms are used for cryptographic key generation, digital signing, and verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

Basic Concepts

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

Plaintext can refer to anything which humans can understand and/or relate to. This may be as simple as English sentences, a script, or Java code. If you can make sense of what is written, then it is in plaintext.

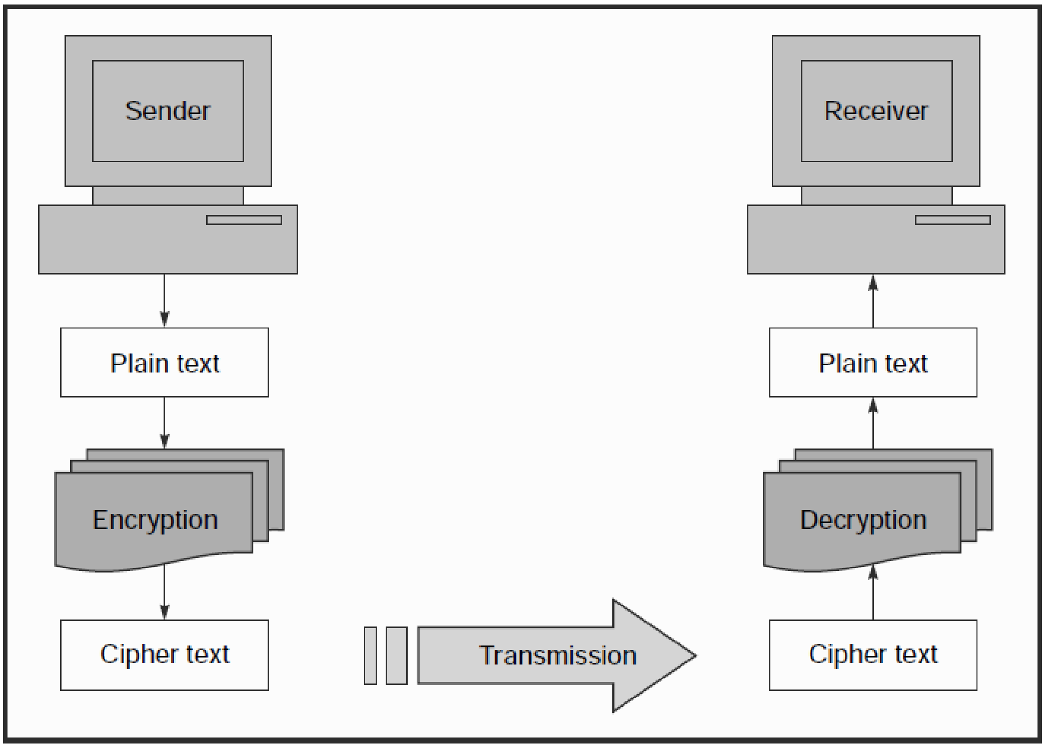
Ciphertext, or encrypted text, is a series of randomized letters and numbers which humans cannot make any sense of. An encryption algorithm takes in a plaintext message, runs the algorithm on the plaintext, and produces a ciphertext. The ciphertext can be reversed through the process of decryption, to produce the original plaintext.



SUBSTITUTION TECHNIQUES

1. CAESAR CIPHER

This was first proposed by **Julius Caesar** and is termed as **Caesar Cipher.** Caesar Cipher is a special case of substitution techniques wherein each alphabetin a message is replaced by an alphabet three places down the line. For instance,using the Caesar Cipher, the plain text ATUL will become cipher text DWXO.



Elements of cryptographic operations

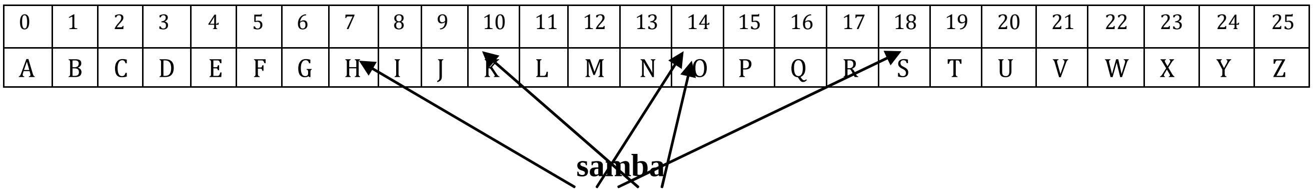
In the **substitution cipher technique**, the characters of a plain text messageare replaced by other *characters, numbers or symbols.*

An attack on a cipher text message, wherein the attacker attempts to use all possible permutations and combinations, is called as a **Bruteforce attack.** The process of trying to break any cipher text message to obtain the original plain text message itself is called as **Cryptanalysis** and the person attempting a cryptanalysis is called as a **cryptanalyst.**

MONO-ALPHABETIC CIPHER

Mono-alphabetic ciphers pose a difficult problem for a cryptanalyst because it can be very difficult to crack thanks to the high number of possible permutations and combinations.

Use random substitution. This means that in a given plain text message**, each A can be replaced by any other alphabet (B through Z), each B can also be replaced by any other random alphabet (A or C through Z) and so on.** The crucial difference being, there is no relation between the replacement of B and replacement of A. That is, if we have decided to replace each A with D, we need not necessarily replace each B with E – we can replace each B with any other character!



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
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

the Cipher text is : <u>*HOSKO*</u>

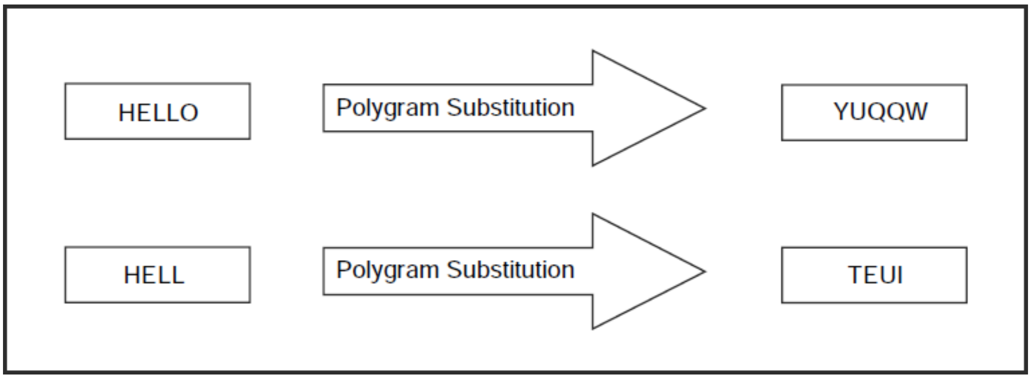
Homophonic Substitution Cipher

Homophonic Substitution Cipher also involves substitution of one plain text character with a cipher text character at a time, however the cipher text character can be any one of the chosen set.

The Homophonic Substitution Cipher is very similar to Mono-alphabetic Cipher. Like a plain substitution cipher technique, we replace one alphabet with another in this scheme. However, the difference between the two techniques is that whereas the replacement alphabet set in case of the simple substitution techniques is fixed (e.g. replace A with D, B with E, etc.), in the case of Homophonic Substitution Cipher**, one plain text alphabet can map to more than one cipher text alphabet**. For instance, A can be replaced by D, H, P, R; B can be replaced by E, I, Q, S, etc.

Polygram Substitution Cipher

In Polygram Substitution Cipher technique, **rather than replacing one plaintext alphabet with one cipher text alphabet at a time, a block of alphabets**is replaced with another block. For instance, HELLO could be replaced byYUQQW, but HELL could be replaced by a totally different cipher text block TEUI, as shown in Fig.



Polyalphabetic Substitution Cipher

A poly-alphabetic cipher is any cipher based on substitution, using several substitution alphabets. In polyalphabetic substitution ciphers, the plaintextletters are enciphered differently based upon their installation in the text.

Rather than being a one-to-one correspondence, there is a one-to-many relationship between each letter and its substitutes.

For example, „a‟ can be enciphered as „d‟ in the starting of the text, but as „n' at the middle. The polyalphabetic ciphers have the benefit of hiding the letter frequency of the basic language. Therefore attacker cannot use individual letterfrequency static to divide the ciphertext.

As the name polyalphabetic recommend this is achieved by **using multiple keys rather than only one key**. This implies that the key should be a stream of subkeys, in which each subkey depends somehow on the position of the plaintextcharacter that needs subkey for encipherment.

Vigenere cipher is one of the simplest and popular algorithms in polyalphabetic cipher. In this approach, the alphabetic text is encrypted using a sequence of **multiple Caesar ciphers** based on the letters of a keyword.

The Vigenère cipher includes several simple substitution ciphers in sequence with several shift values. In this cipher, the keyword is repeated just before it connects with the duration of the plaintext.

Encryption Process:

$$C_i=(P_i+K_i) \bmod 26$$

In this process sum of ith position of plain text and ith position of key will be added and applied modulus 26 on the result , the generated positional value will be considered as Cipher text.

Decryption Process

$$P_i=(C_i-K_i) \bmod 26$$

In this process sum of ith position of Cipher text and ith position of key will be subtracted and applied modulus 26 on the result , the generated positional valuewill be considered as Plain text.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
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

Key: samba

Plain text: hello students how are youCipher
text:

Key	s	a	m	B	a	s	a	m	b	a	s	a	m	b	a	s	a	m	b	a	s	A
PT	h	e	l	L	o	s	t	u	d	e	n	t	s	h	o	w	a	r	e	y	o	U
CT	z	e	x	M	o	k	t	g	E	e	f	t	e	i	o	o	a	c	f	y	g	U

Apply Encryption process to generate cipher text

That is „s" position is 18 and „h" position is 7 so now C_1=(p_1+k_1)mod26

= (18+7)mpd26

=(25)mod26

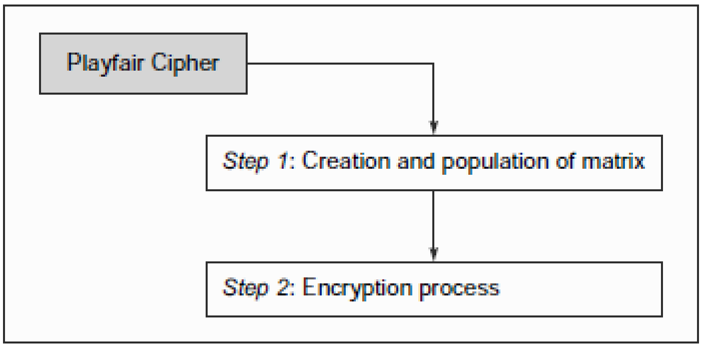
=25 (which is equalent to „z")

Like wise generate the table accordingly.

Playfair Cipher:

The Playfair Cipher, also called as **Playfair Square**, is a cryptographic techniquethat is used for manual encryption of data.

The Playfair encryption scheme uses two main processes, as shown in Fig


Playfair Cipher
Step 1: Creation and population of matrix
Step 2: Encryption process

Playfair cipher steps

Step 1: Creation Population of Matrix and The Playfair Cipher makes use of a 5 x 5 matrix (table), which is used to store a keyword or phrase that becomes the key for encryption and decryption.

The way this is entered into the 5 x 5 matrix is based on some simple rules, asshown below

1. Enter the keyword in the matrix row-wise: left-to-right, and then top-to-bottom.
2. Drop duplicate letters.
3. Fill the remaining spaces in the matrix with the rest of the English alphabets (A-Z) that were not a part of our keyword. While doing so, combine I and J in the same cell of the table. In other words, if I or J is a part of the keyword, disregard both I and J while filling the remaining slots.

Algorithm to encrypt the plain text: The plaintext is split into pairs of twoletters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

For example:

PlainText: "instruments"

After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

1. Pair cannot be made with same letter. Break the letter in single and add abogus letter to the previous letter.

Plain Text: “hello”

After Split: „he' „lx' „lo'Here **‘x’** is the bogus letter.

2. If the letter is standing alone in the process of pairing, then add an extrabogus letter with the alone letter

Plain Text: “helloe” **AfterSplit:** „he' „lx' „lo' „ez'Here **‘z’** is the bogus letter.**Rules for Encryption:**

- **If both the letters are in the same column**: Take the letter below eachone (going back to the top if at the bottom).

  For example:

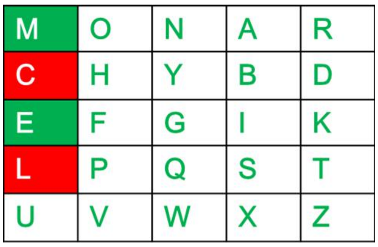
Diagraph: "me"

Encrypted Text: cl

Encryption:

m -> c

e -> l



M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If both the letters are in the same row: Take the letter to the right of eachone (going back to the leftmost if at the rightmost position).

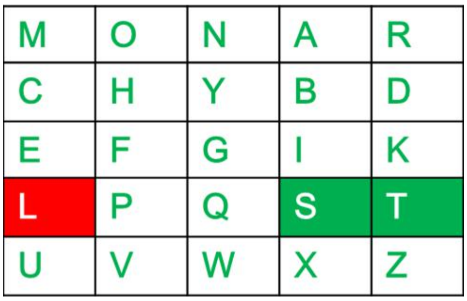
Diagraph: "ST"

Encrypted Text: TL

Encryption:

S -> T

T -> L



M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If neither of the above rules is true: Form a rectangle with the two lettersand take the letters on the horizontal opposite corner of the rectangle.

For example:

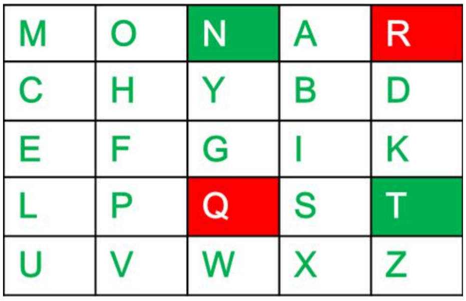
Diagraph: "nt"

Encrypted Text: rq

Encryption:

n -> r

t -> q



M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Plain Text: "instrumentsz" Encrypted Text: gatlmzclrqtxEncryption:

i -> g

n -> a

s -> t t

-> l r -

> mu -

> zm -

> ce -

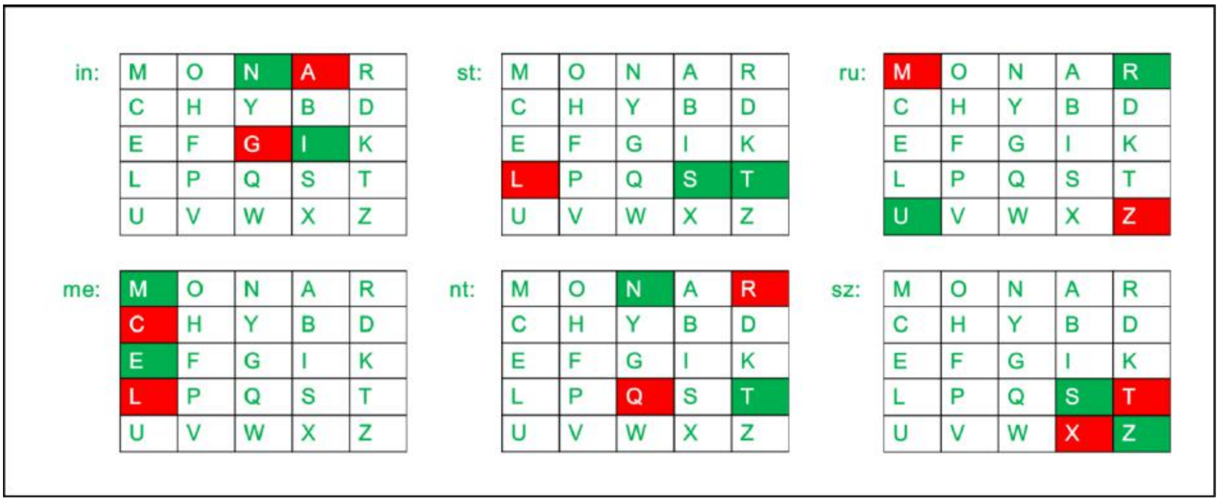
> l n -

> r t -

> q s -

> t z -

> x



in:

M	O	🟥 N	🟥 A	R
C	H	Y	B	D
E	F	🟥 G	🟩 I	K
L	P	Q	S	T
U	V	W	X	Z

st:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
🟥 L	P	Q	🟩 S	🟩 T
U	V	W	X	Z

ru:

🟥 M	O	N	A	🟩 R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
🟩 U	V	W	X	🟥 Z

me:

🟩 M	O	N	A	R
🟥 C	H	Y	B	D
🟩 E	F	G	I	K
🟥 L	P	Q	S	T
U	V	W	X	Z

nt:

M	O	🟩 N	A	🟥 R
C	H	Y	B	D
E	F	G	I	K
L	P	🟥 Q	S	🟩 T
U	V	W	X	Z

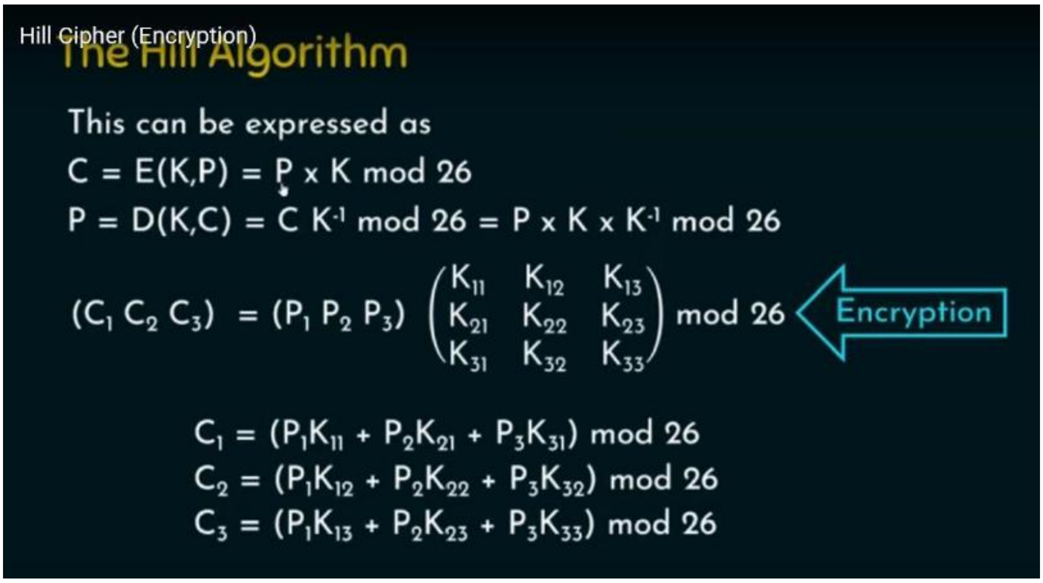
sz:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	🟩 S	🟥 T
U	V	W	🟥 X	🟩 Z

Hill Cipher

Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0,B= 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).



Hill Cipher (Encryption)

The Hill Algorithm

This can be expressed as

$C = E(K,P) = P \times K \bmod 26$

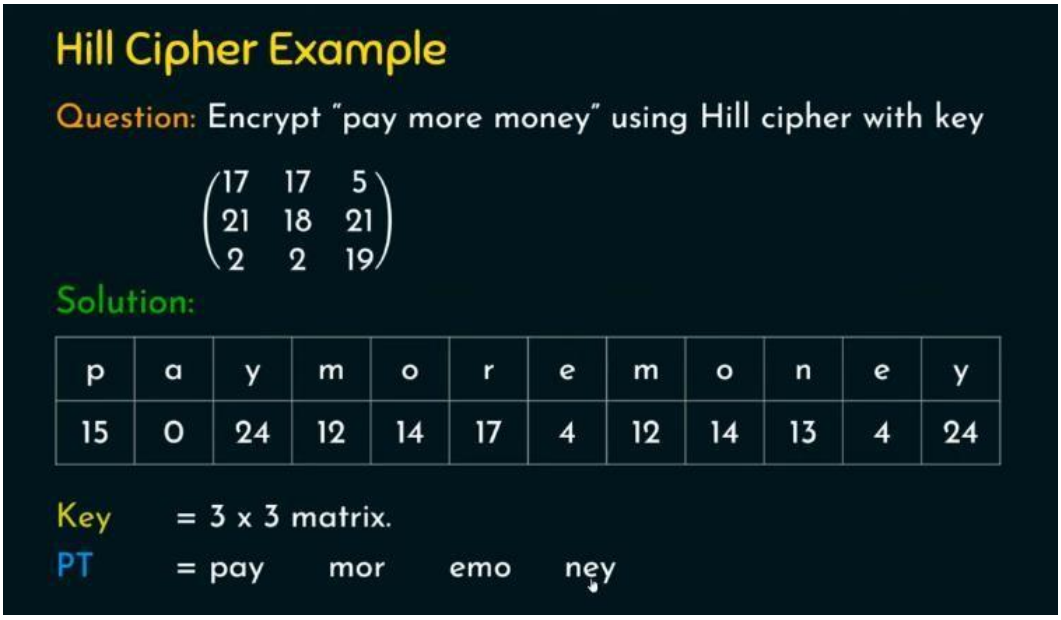
$P = D(K,C) = C\,K^{-1} \bmod 26 = P \times K \times K^{-1} \bmod 26$

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

$C_1 = (P_1K_{11} + P_2K_{21} + P_3K_{31}) \bmod 26$

$C_2 = (P_1K_{12} + P_2K_{22} + P_3K_{32}) \bmod 26$

$C_3 = (P_1K_{13} + P_2K_{23} + P_3K_{33}) \bmod 26$



Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

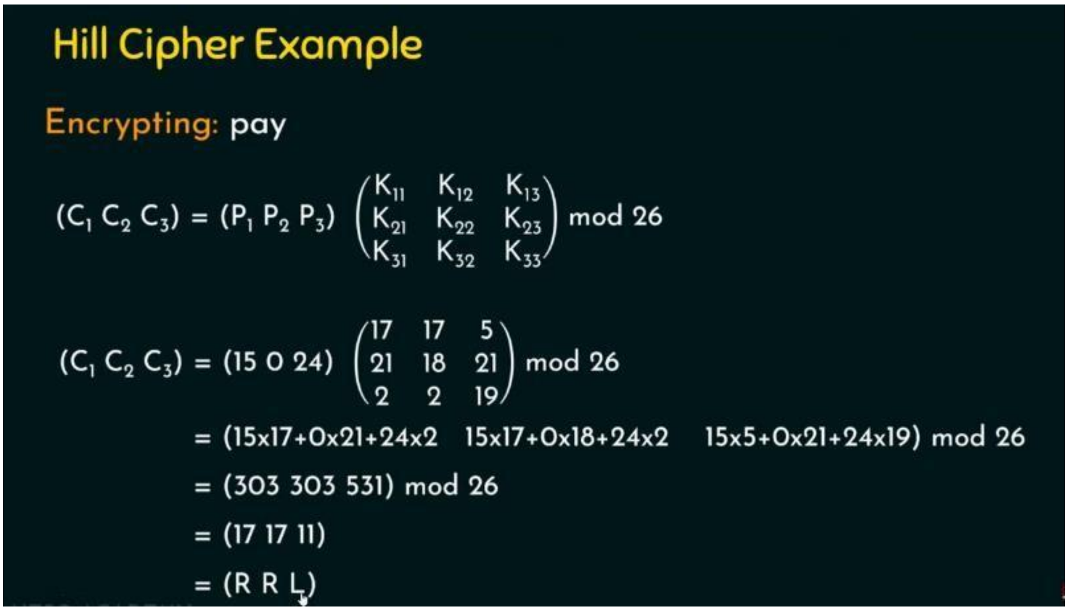
$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

p	a	y	m	o	r	e	m	o	n	e	y
15	0	24	12	14	17	4	12	14	13	4	24

Key = 3 x 3 matrix.

PT = pay mor emo ney



Hill Cipher Example

Encrypting: pay

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

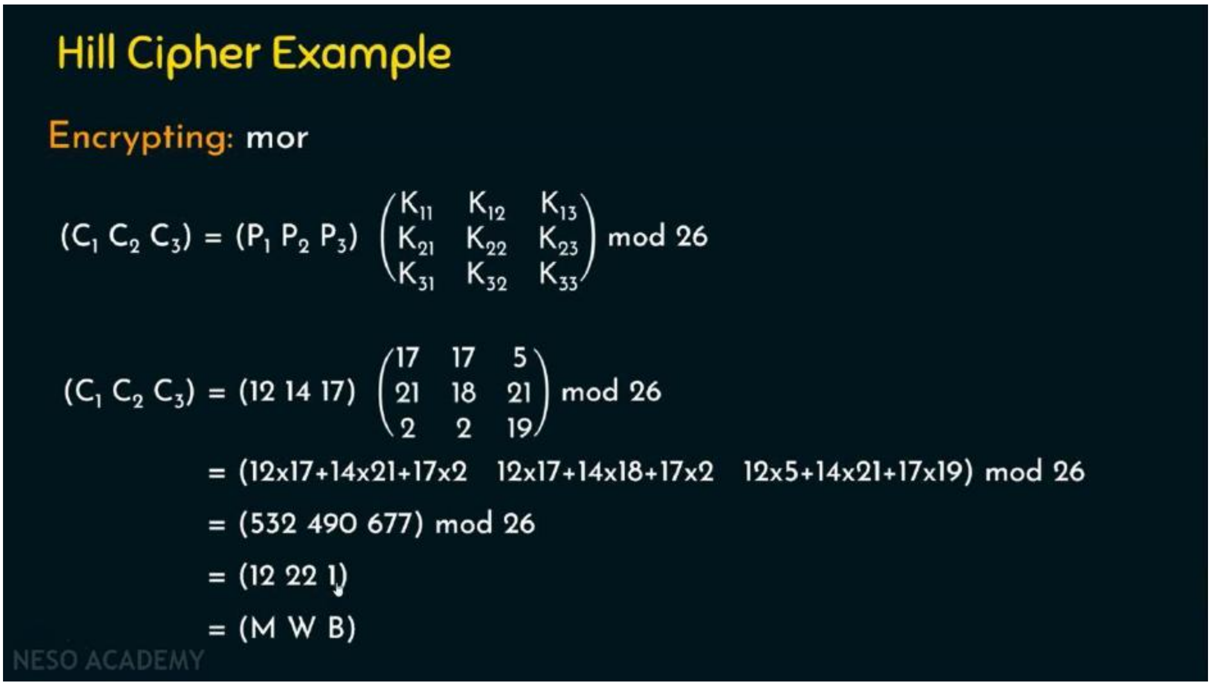
$$(C_1\ C_2\ C_3) = (15\ 0\ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (15x17+0x21+24x2 \quad 15x17+0x18+24x2 \quad 15x5+0x21+24x19) \bmod 26$$

$$= (303\ 303\ 531) \bmod 26$$

$$= (17\ 17\ 11)$$

$$= (R\ R\ L)$$



Hill Cipher Example

Encrypting: mor

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

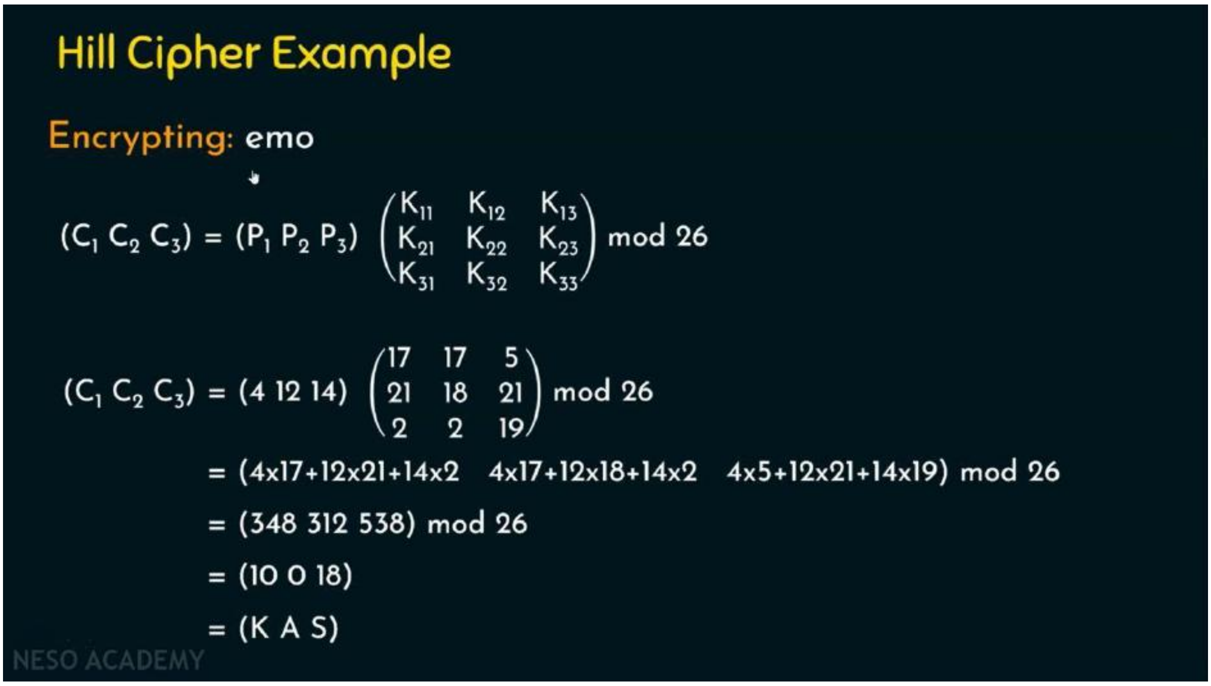
$$(C_1\ C_2\ C_3) = (12\ 14\ 17) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (12x17+14x21+17x2 \quad 12x17+14x18+17x2 \quad 12x5+14x21+17x19) \bmod 26$$

$$= (532\ 490\ 677) \bmod 26$$

$$= (12\ 22\ 1)$$

$$= (M\ W\ B)$$



Hill Cipher Example

Encrypting: emo

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3)\begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

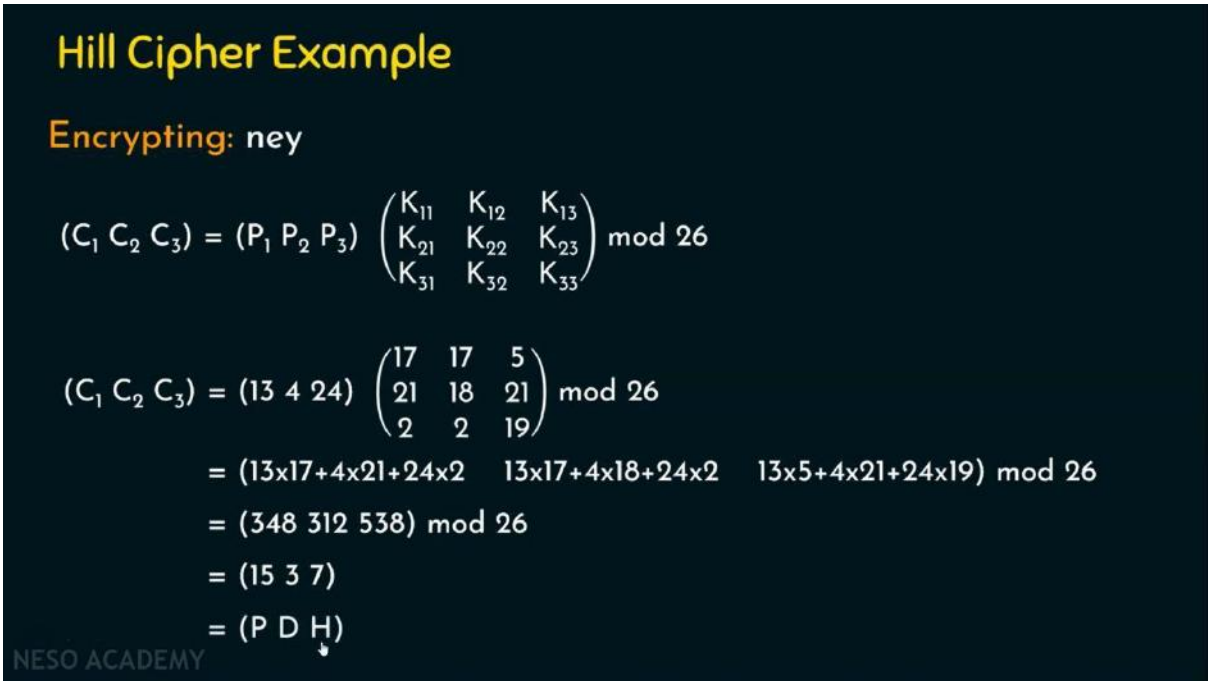
$$(C_1\ C_2\ C_3) = (4\ 12\ 14)\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (4\text{x}17+12\text{x}21+14\text{x}2 \quad 4\text{x}17+12\text{x}18+14\text{x}2 \quad 4\text{x}5+12\text{x}21+14\text{x}19) \bmod 26$$

$$= (348\ 312\ 538) \bmod 26$$

$$= (10\ 0\ 18)$$

$$= (K\ A\ S)$$



Hill Cipher Example

Encrypting: ney

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3)\begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

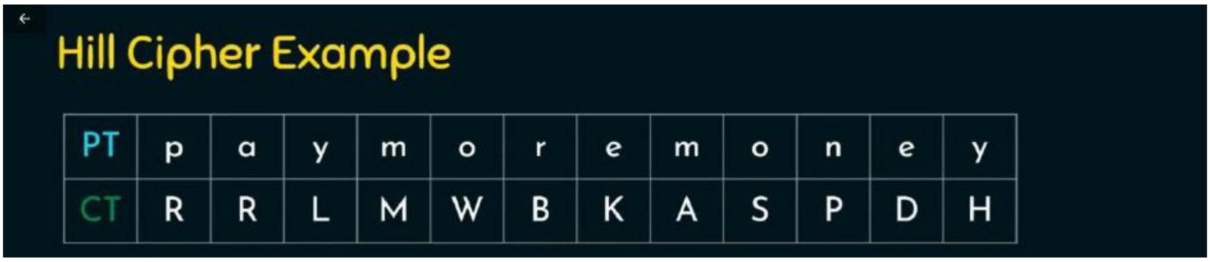
$$(C_1\ C_2\ C_3) = (13\ 4\ 24)\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (13\text{x}17+4\text{x}21+24\text{x}2 \quad 13\text{x}17+4\text{x}18+24\text{x}2 \quad 13\text{x}5+4\text{x}21+24\text{x}19) \bmod 26$$

$$= (348\ 312\ 538) \bmod 26$$

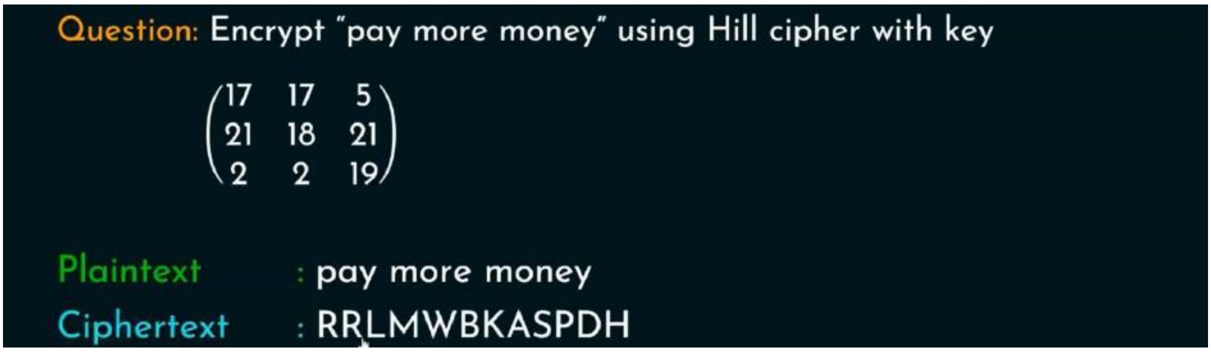
$$= (15\ 3\ 7)$$

$$= (P\ D\ H)$$



Hill Cipher Example

PT	p	a	y	m	o	r	e	m	o	n	e	y
CT	R	R	L	M	W	B	K	A	S	P	D	H



Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Plaintext : pay more money

Ciphertext : RRLMWBKASPDH

TRANSPOSITION TECHNIQUES

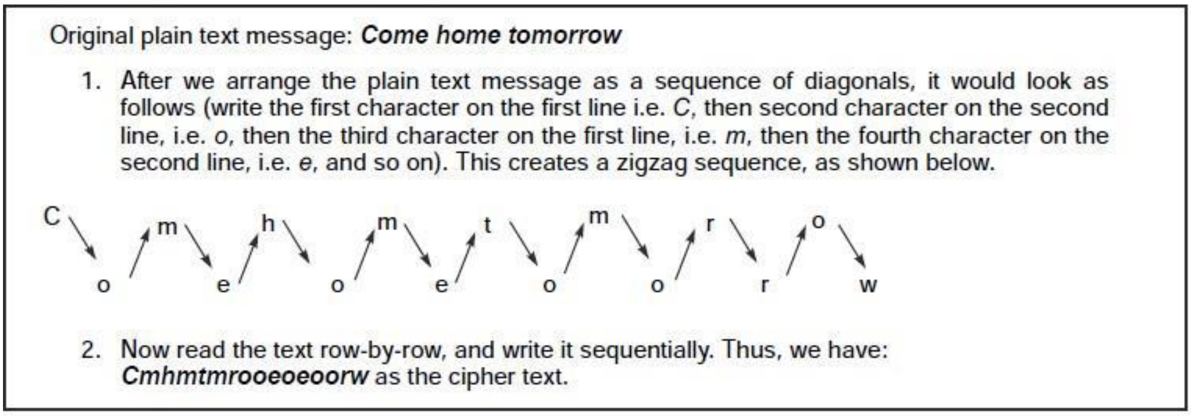
Transposition techniques differ from substitution techniques in the way that they do not simply replace one alphabet with another: they also perform some permutation over the plain text alphabets.

Rail Fence Technique

Rail fence technique involves writing plain text as sequence of diagonals andthen reading it row-by-row to produce cipher text.

Suppose that we have a plain text message Come home tomorrow. How would we transform that into a cipher text message using the Rail Fence Technique? This is shown in Fig.

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in Step 1 as a sequence of rows.
3. here depth=2



Original plain text message: ***Come home tomorrow***

1. After we arrange the plain text message as a sequence of diagonals, it would look as follows (write the first character on the first line i.e. *C*, then second character on the second line, i.e. *o*, then the third character on the first line, i.e. *m*, then the fourth character on the second line, i.e. *e*, and so on). This creates a zigzag sequence, as shown below.

2. Now read the text row-by-row, and write it sequentially. Thus, we have: ***Cmhmtmrooeoeoorw*** as the cipher text.

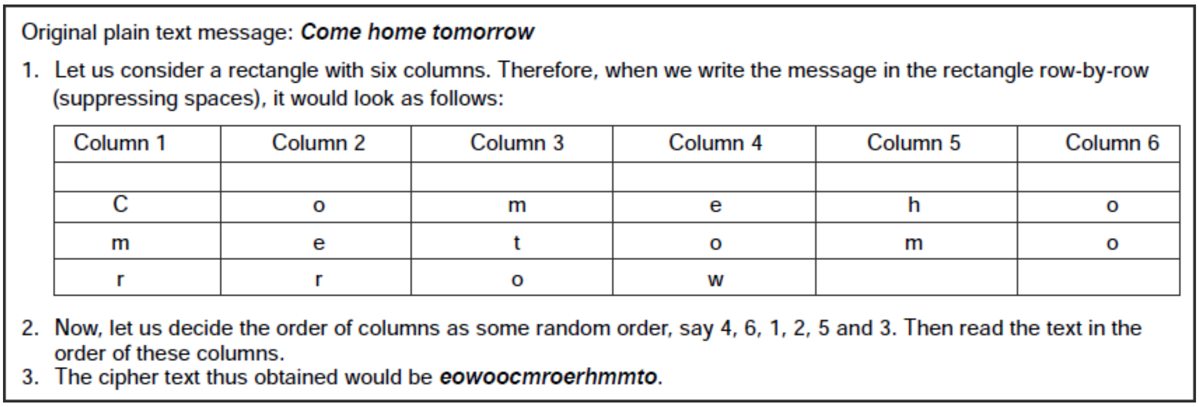
Example of rail technique

Simple Columnar Transposition Technique

Basic Technique Variations of the basic transposition technique such as Rail Fence Technique exist. Such a scheme call as Simple Columnar TranspositionTechnique.

- Write the plain text message row-by-row in a rectangle of a pre-definedsize.
- Read the message column-by-column. However, it need not be in the orderof columns 1,2, 3 etc. It can be any random order such as 2, 3, 1, etc.
- The message thus obtained is the cipher text message.

The Simple Columnar Transposition Technique simply arranges the plain text asa sequence of rows of a rectangle that are read in columns randomly.



Original plain text message: ***Come home tomorrow***

1. Let us consider a rectangle with six columns. Therefore, when we write the message in the rectangle row-by-row (suppressing spaces), it would look as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
C	o	m	e	h	o
m	e	t	o	m	o
r	r	o	w		

2. Now, let us decide the order of columns as some random order, say 4, 6, 1, 2, 5 and 3. Then read the text in the order of these columns.
3. The cipher text thus obtained would be ***eowoocmroerhmmto***.

Example of simple columnar technique

E-mail Security

PRETTY GOOD PRIVACY

PGP is an open-source, freely available software package for e-mail security. It provides authentication through the use of digital signature, confidentiality through the use of symmetric block encryption, compression using the ZIP algorithm, and e-mail compatibility using the radix-64 encoding scheme.

Notations:

K_s = session key used in symmetric encryption scheme
PR_a = private key of user A, used in public-key encryption schemePU_a
= public key of user A, used in public-key encryption scheme EP
= public-key encryption
DP = public-key decryption EC =
symmetric encryption DC =
symmetric decryption H = hash
function

|| = concatenation

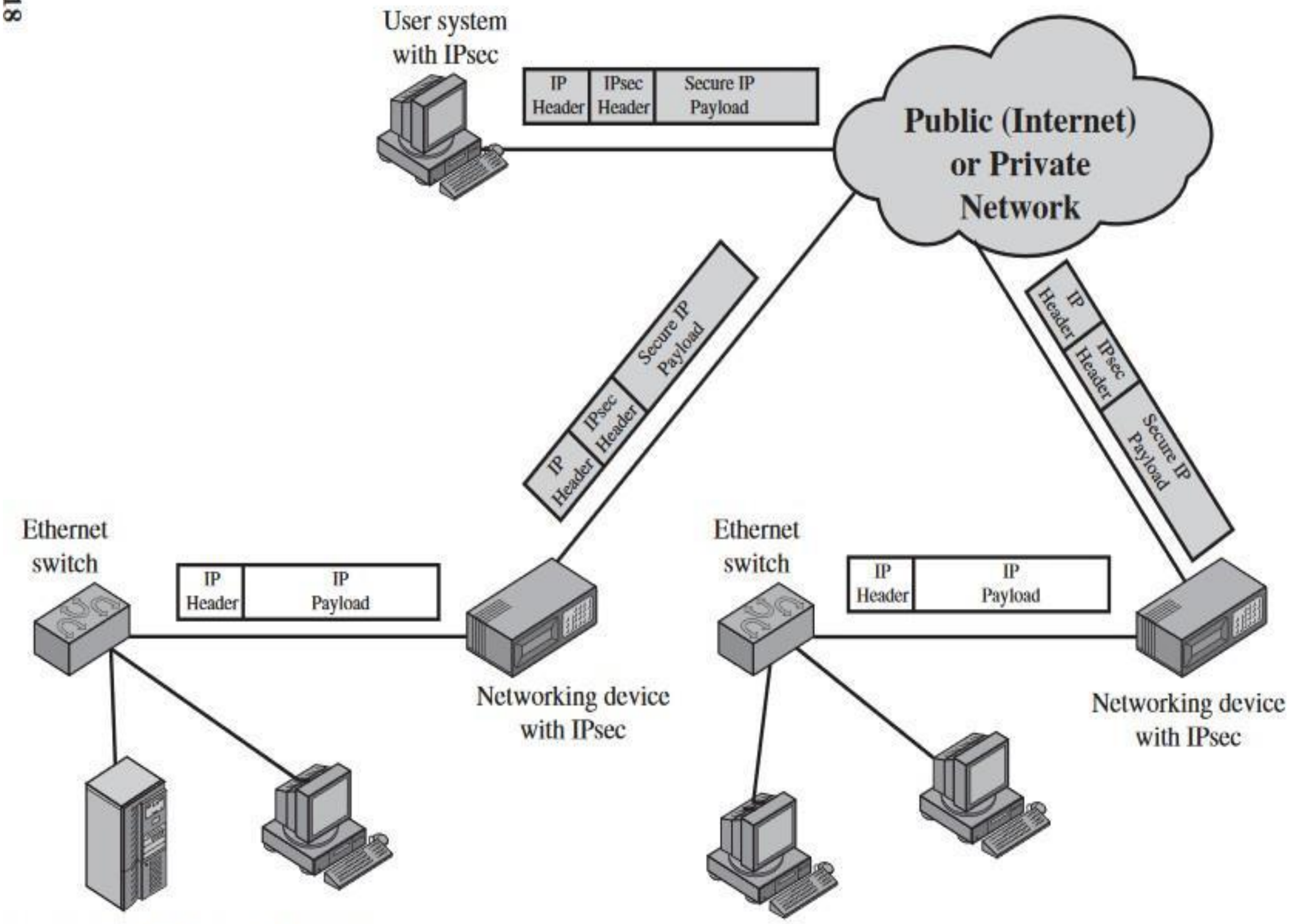
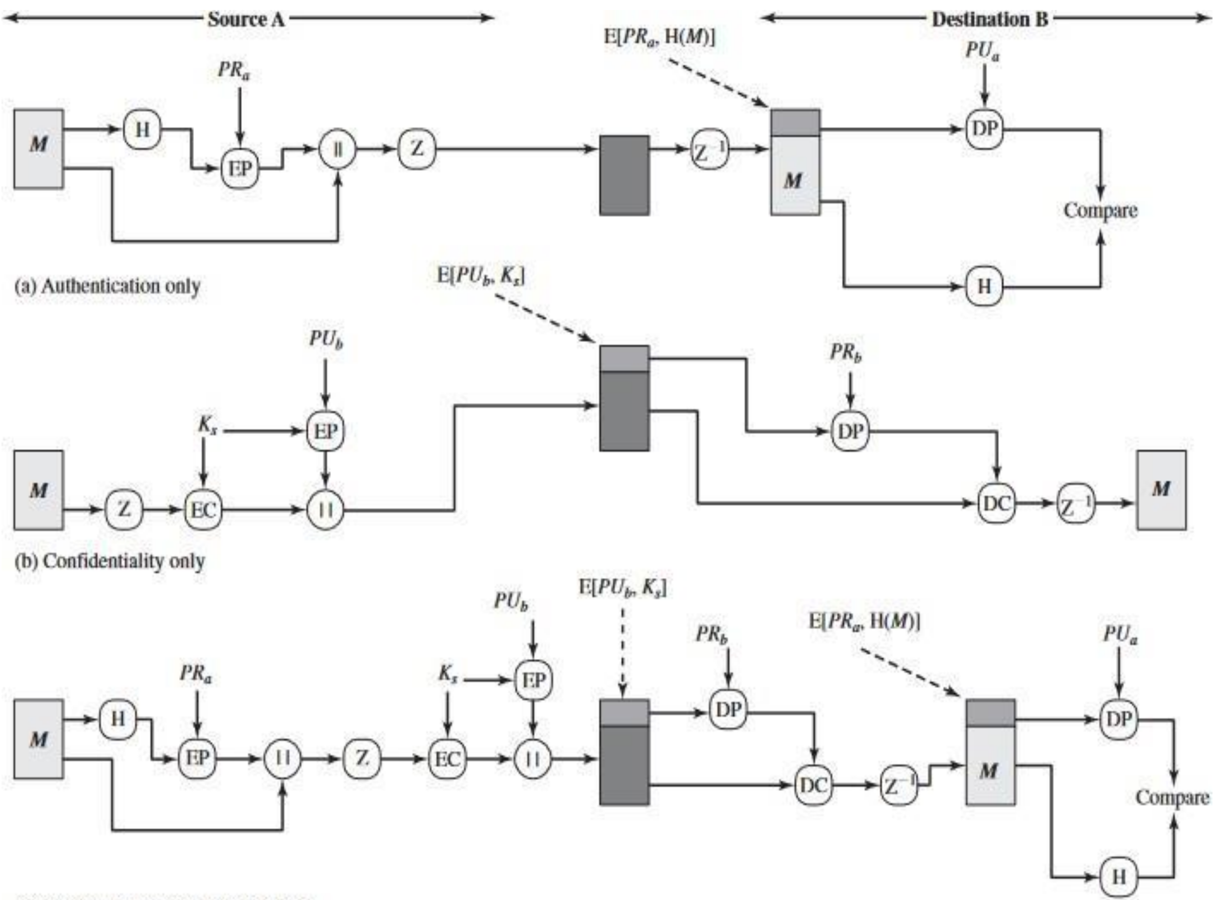


Figure 19.1 An IP Security Scenario

Z = compression using ZIP algorithm

R64 = conversion to radix 64 ASCII format



(a) Authentication only

(b) Confidentiality only

(c) Confidentiality and authentication

Operational description:

The actual operation of PGP, as opposed to the management of keys, consists of four services: authentication, confidentiality, compression, and e-mail compatibility.

Authentication:

The sequence of steps as follows

1. The sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender‟s private key, and the result is prepended to the message.
4. The receiver uses RSA with the sender‟‟s public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is acceptedas authentic.

CONFIDENTIALITY Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to be stored locally as files

The sequence is as follows.

1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.

2. The message is encrypted using CAST-128 (or IDEA or 3DES) with thesession key.
3. The session key is encrypted with RSA using the recipient‟s public keyand is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover thesession key.
5. The session key is used to decrypt the message.

CONFIDENTIALITY AND AUTHENTICATION As both services may be used for the same message.

- First, a signature is generated for the plaintext message and prepended to the message.
- Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).
- This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message.
- It is generally more convenient to store a signature with a plaintext version of a message.
- Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.

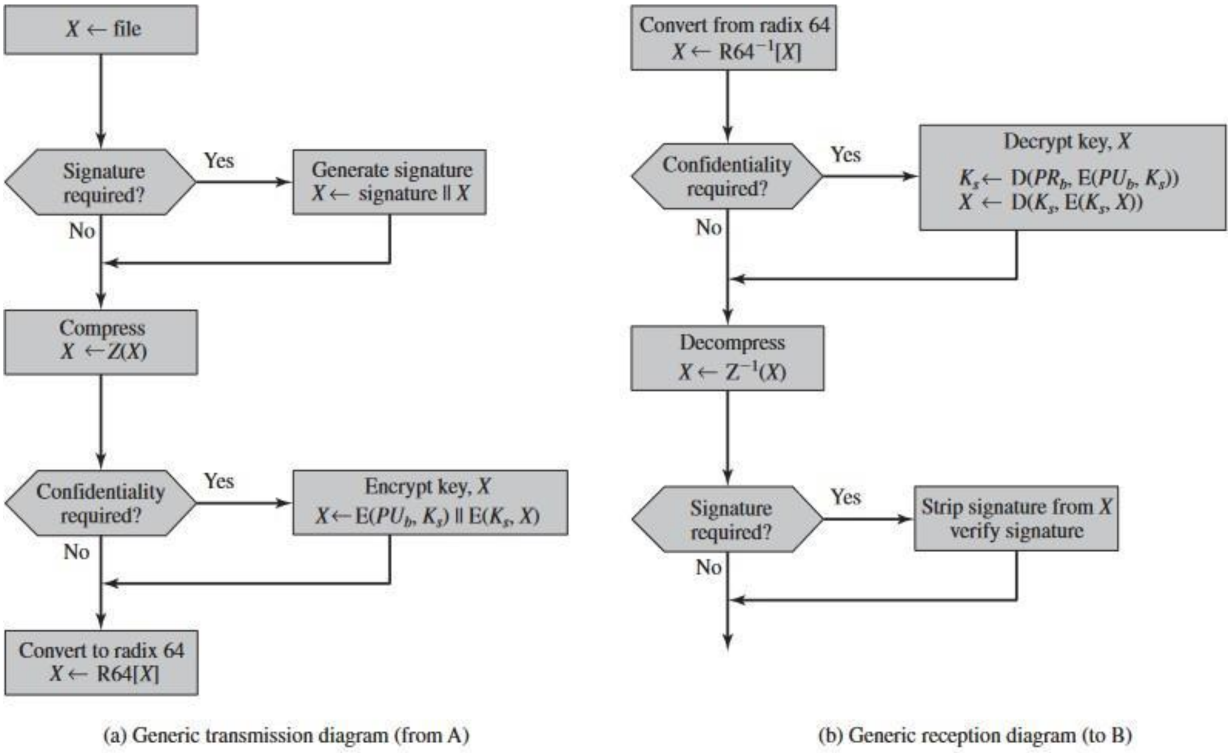
COMPRESSION As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.

1. The signature is generated before compression for two reasons:
   - It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.
   - Even if one were willing to generate dynamically a recompressed message for verification, PGP‟s compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms.
2. Message encryption is applied after compression to strengthen cryptographicsecurity.

E-MAIL COMPATIBILITY When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender‟s private key).

- If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key).
- Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets.

- However, many electronic mail systems only permit the use of blocksconsisting of ASCII text.
- To accommodate this restriction, PGP provides the service of convertingthe raw 8-bit binary stream to a stream of printable ASCII characters.



$X \leftarrow$ file
Signature required?
Yes
Generate signature $X \leftarrow$ signature ‖ X
No
Compress $X \leftarrow Z(X)$
Confidentiality required?
Yes
Encrypt key, X $X \leftarrow E(PU_b, K_s) \| E(K_s, X)$
No
Convert to radix 64 $X \leftarrow R64[X]$
Convert from radix 64 $X \leftarrow R64^{-1}[X]$
Confidentiality required?
Yes
Decrypt key, X $K_s \leftarrow D(PR_b, E(PU_b, K_s))$ $X \leftarrow D(K_s, E(K_s, X))$
No
Decompress $X \leftarrow Z^{-1}(X)$
Signature required?
Yes
Strip signature from X verify signature
No
(a) Generic transmission diagram (from A)
(b) Generic reception diagram (to B)

Figure 18.2 Transmission and Reception of PGP Messages

PGP Message Generation:

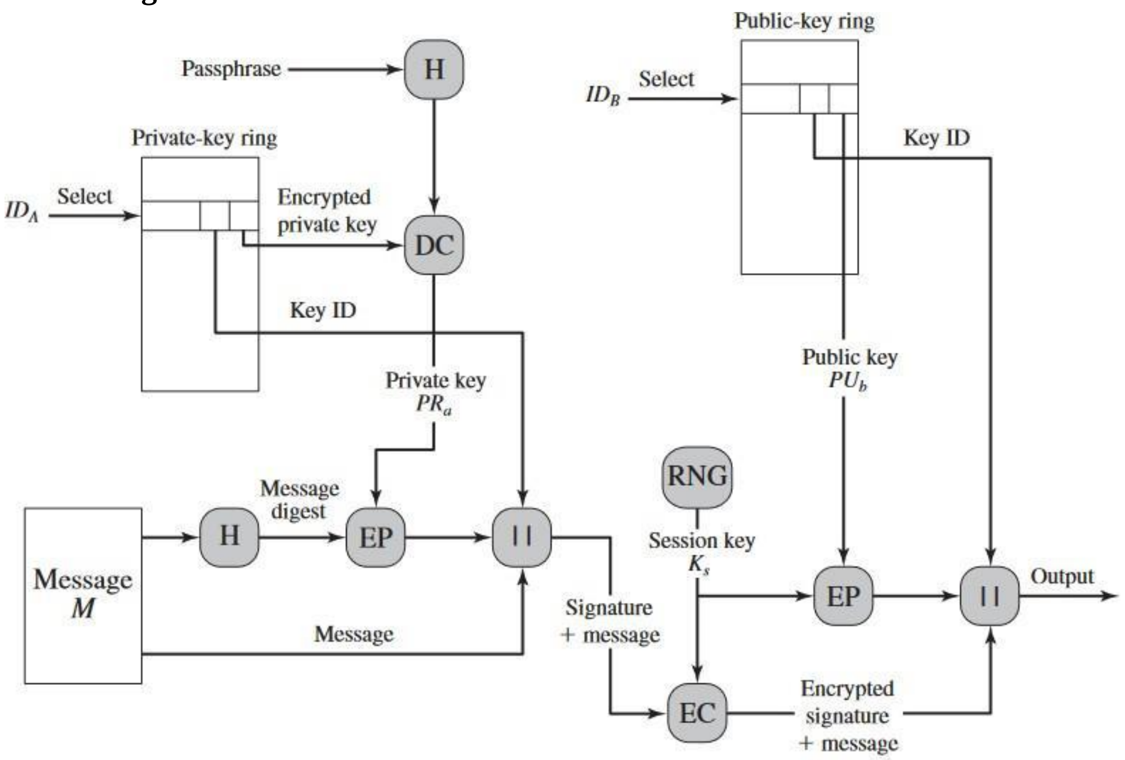


Figure 18.5 PGP Message Generation (from User A to User B: no compression or radix-64 conversion)

S/MIME (Secure/Multipurpose Internet Mail Extensions)

S/MIME is a security enhancement to the MIME Internet e-mail formatstandard based on technology from RSA Data Security.

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations ofthe use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or someother mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] liststhe following limitations of the SMTP/5322 scheme.

- SMTP cannot transmit executable files or other binary objects.
- SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
- SMTP servers may reject mail message over a certain size.
- SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
- SMTP gateways to X.400 electronic mail networks cannot handle non- textual data included in X.400 messages.

Header fields in MIME:

The five header fields defined in MIME are

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body;this is useful when the object is not readable (e.g., audio data).

IP SECURITY OVERVIEW

IP security (IPsec) is a capability that can be added to either current version of the Internet Protocol (IPv4 or IPv6) by means of additional headers. IPsec encompasses three functional areas: authentication, confidentiality, and key management.

In 1994, the Internet Architecture Board (IAB) issued a report titled “Security in the Internet Architecture”

To provide security, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6.

Applications of IPsec

IPsec provides the capability to secure communications across a LAN, acrossprivate and public WANs, and across the Internet. Examples of its use include: **Secure branch office connectivity over the Internet**: A company can build a secure virtual private network over the Internet or over a public WAN.

Secure remote access over the Internet: An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network.

Establishing extranet and intranet connectivity with partners: IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

Enhancing electronic commerce security: Even though some Web and electronic commerce applications have built-in security protocols, the use of IPsec enhances that security.

The principal feature of IPsec that enables it to support these varied applications is that it can encrypt and/or authenticate all traffic at the IP level.Thus, all distributed applications (including remote logon, client/server, e-mail, file transfer, Web access, and so on) can be secured.

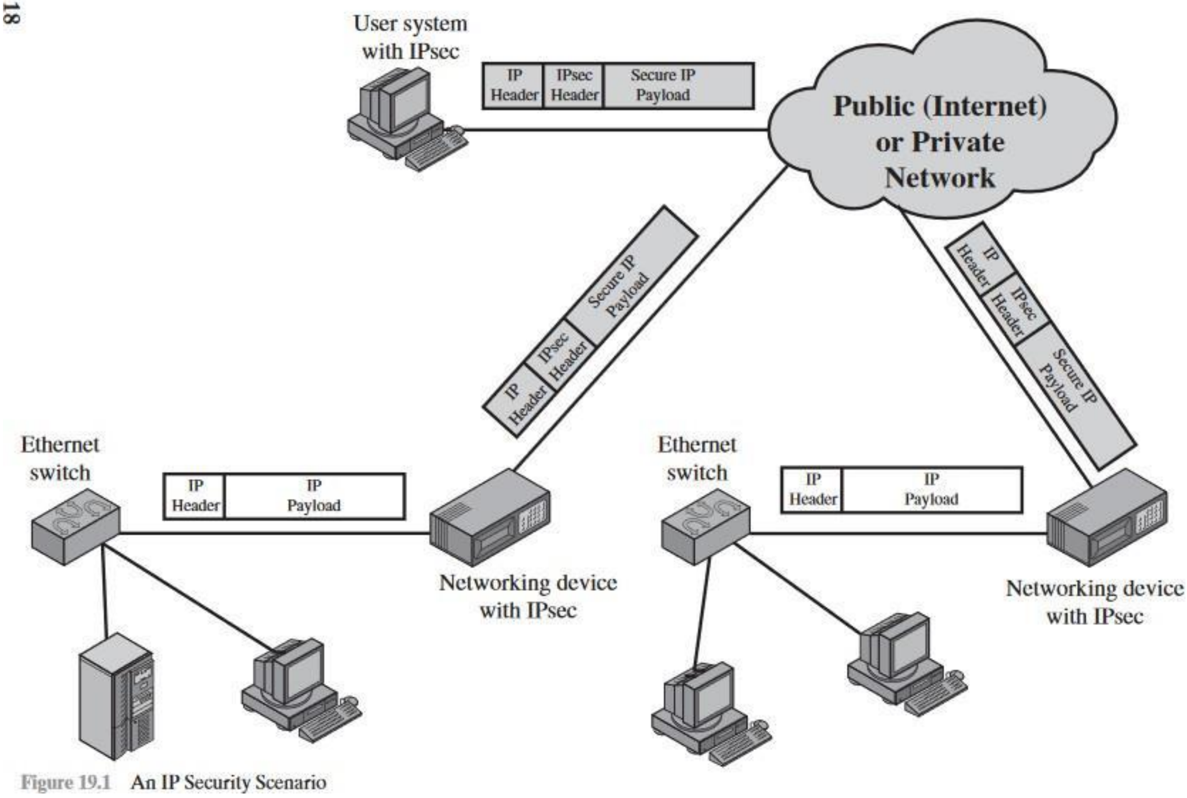


Figure 19.1 An IP Security Scenario

Figure 19.1 is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Non-secure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used. These protocols operate in networking devices, such as a router or firewall, that connect each LAN to the outside world.

Benefits of IPsec

Some of the benefits of IPsec:

- When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security- related processing.
- IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.
- IPsec is below the transport layer (TCP, UDP) and so is transparent to applications.
- IPsec can be transparent to end users.
- IPsec can provide security for individual users if needed.

Routing Applications

- Router advertisement (a new router advertises its presence) comes from an authorized router.
- A neighbor advertisement (a router seeks to establish or maintain a neighbour relationship with a router in another routing domain) comes from an authorized router.
- A redirect message comes from the router to which the initial IP packet was sent.
- A routing update is not forged.

IPsec Services

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IP SECURITY ARCHITECTURE

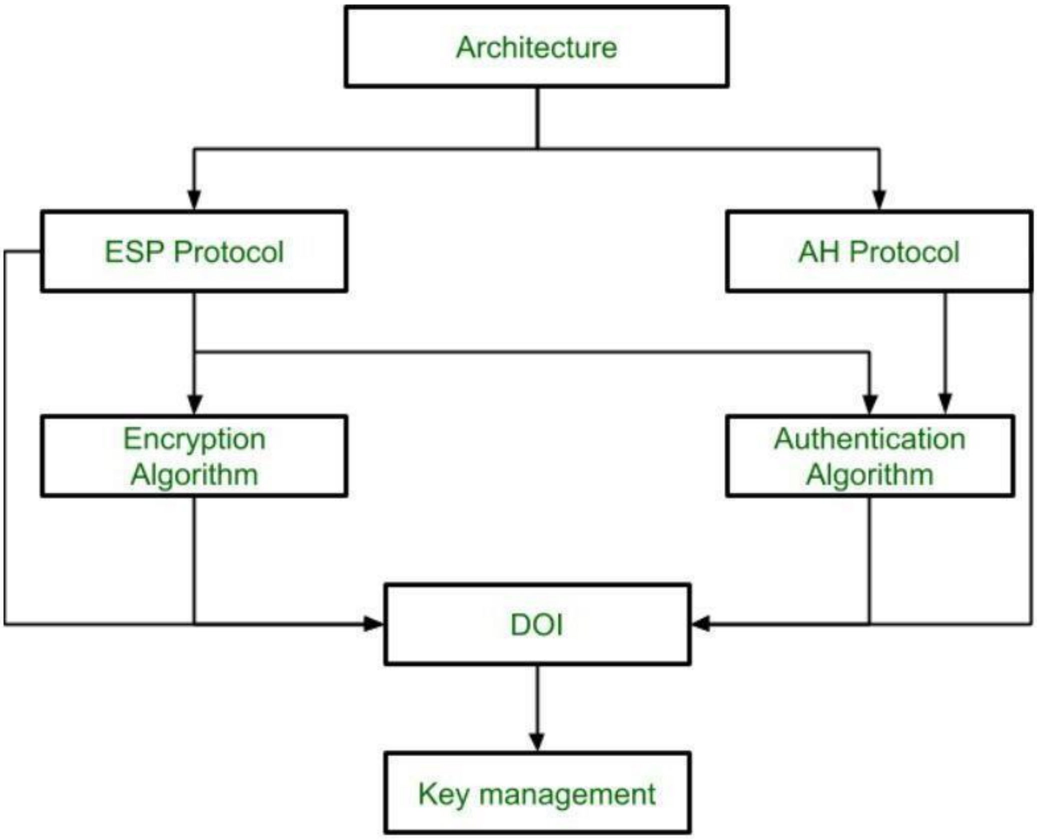
IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture includes protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

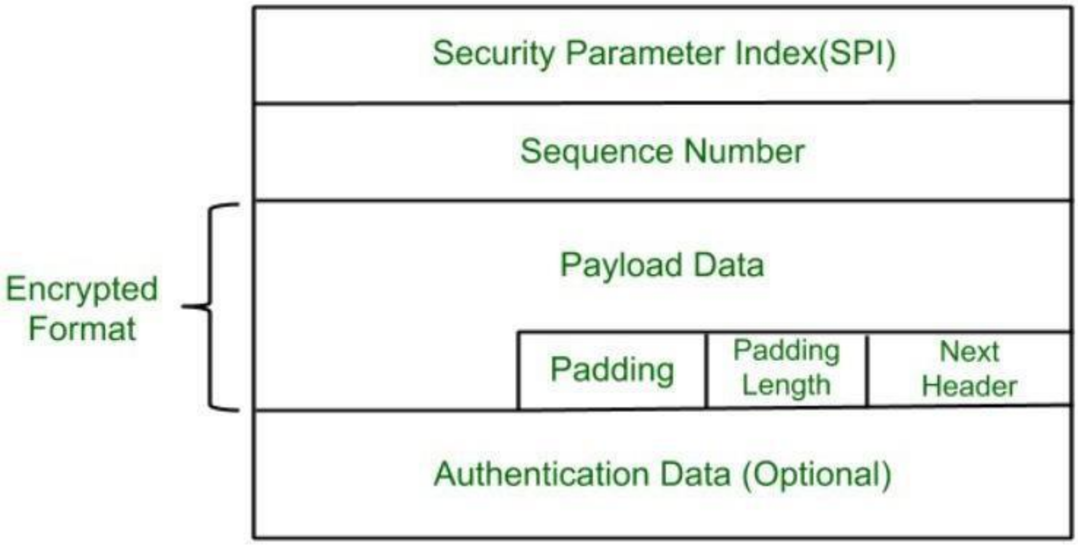
１. **Architecture:** Architecture or IP Security Architecture covers the generalconcepts, definitions, protocols, algorithms, and security requirements of IP Security technology.

２. **ESP Protocol:** ESP(Encapsulation Security Payload) provides a confidentialityservice. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.



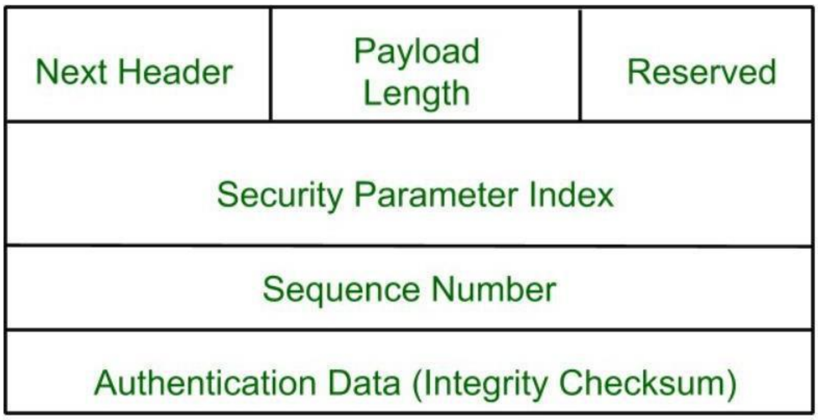
Packet Format:



- **Security Parameter Index(SPI):** This parameter is used by Security Association. It is used to give a unique number to the connection built between the Client and Server.
- **Sequence Number:** Unique Sequence numbers are allotted to every packet so that on the receiver side packets can be arranged properly.
- **Payload Data:** Payload data means the actual data or the actual message. The Payload data is in an encrypted format to achieve confidentiality.
- **Padding:** Extra bits of space are added to the original message in order to ensure confidentiality. Padding length is the size of the added bits of space in the original message.
- **Next Header:** Next header means the next payload or next actual data.
- **Authentication Data** This field is optional in ESP protocol packet format.

3. Encryption algorithm: The encryption algorithm is the document that describes various encryption algorithms used for Encapsulation Security Payload.

4. AH Protocol: AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.



Authentication Header covers the packet format and general issues related to the use of AH for packet authentication and integrity.

5. **Authentication Algorithm:** The authentication Algorithm contains the set ofdocuments that describe the authentication algorithm used for AH and for the authentication option of ESP.

6. **DOI (Domain of Interpretation):** DOI is the identifier that supports both AH and ESP protocols. It contains values needed for documentation related to each other.

7. **Key Management:** Key Management contains the document that describes how the keys are exchanged between sender and receiver.