COMPUTER NETWORKS

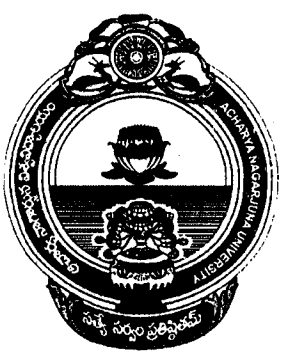
M.Sc. Information Technology , I year, Paper - II

Lesson Writer

Dr. P. Ammi Reddy, MSc., MCA, M.Tech., Ph.D.
Associate Professor
Vasireddy Venkatadri Institute of Technology
Nambur, Guntur - Dist

Editor

Prof. E.Sreenivasa Reddy, B.Tech.,M.Tech., M.S., Ph.D.,
Principal
ANU college of Engineering & Technology
Acharya Nagarjuna University



Director

Prof. P. VARA PRASADA MURTHY
M.A. Sahitya Acharya, Visaradha (Hindi), Vidhya Varadhi (Ph.D)
PROFESSOR IN SANSKRIT
Head, Department of Telugu & Oriental Languages

Centre for Distance Eduction
Acharya Nagarjuna University
Nagarjuna Nagar-522510

Ph: 0863-2293299, 2293356, 08645-211023, Cell:98482 85518
0863 - 2346259 (Study Material)
Website : www.anucde.ac.in or www.anucde.info
e-mail : info@ anucde.ac.in

M.Sc. (IT) : Computer Networks

Edition : 2019

No. of Copies : 120

Published by :

Prof. P. Vara Prasada Murthy,
Director
Centre for Distance Education,
Acharya Nagarjuna University

Printed at :

M/s. Romith Technologies
Guntur.

FOREWORD

Acharya Nagarjuna University, since its establishment in 1976, has been moving ahead in the path of academic excellence, offering a variety of courses and research contributions. The University achieved recognition as one of the eminent universities in the country by gaining A grade from the NAAC 2016. At present Acharya Nagarjuna University is offering educational opportunities at the UG, PG levels to students of 447 affiliated colleges spread over the two districts of Guntur and Prakasam.

The University had started the Centre for Distance Education in 2003-04 with the aim to bring Higher education within the reach of all. The Centre has been extending services to those who cannot join in colleges, cannot afford the exorbitant fees as regular students, and to housewives desirous of pursuing higher studies to study B.A., B.Com, and B.Sc., Courses at the Degree level and M.A., M.Com., M.Sc, M.B.A. and LL.M. courses at the PG level.

For better understanding by students, self-instruction materials have been prepared by eminent and experienced teachers. The lessons have been prepared with care and expertise. However constructive ideas and scholarly suggestions are welcome from students and teachers. Such ideas will be incorporated for the greater efficacy of the distance mode of education. For clarification of doubts and feedback, Weekly classes and contact classes are arranged at UG and PG levels respectively.

I wish the students who pursue higher education through Centre for Distance Education will not only be personally benefited by improving their qualifications but also strive for nation's growth by being a member in Knowledge society I hope that in the years to come, the Centre for Distance Education will grow in strength by introducing new courses, catering to the needs of people. I congratulate all the Directors, Academic coordinators, Editors, Lesson - Writers, and Academic Counsellors and Non-teaching staff of the Centre who have been extending their services in these endeavours.

Prof. KOONA RAMJI
Vice-Chancellor I/C
Acharya Nagarjuna University

M.Sc. (IT) , I year

Syllabus

Paper - II : COMPUTER NETWORKING

UNIT – I :

Data Communications, Communications Networks

UNIT – II :

Network Technologies, Multiple Access

UNIT – III:

Switching, Naming and Addressing

UNIT – IV:

Routing, Services & Applications

UNIT – V :

Security, Binary Arithmetic & IP address calculation.

Prescribed Book:

1. Tittle ED; Computer Networking: Schaum's outlines (TMH),

2. Chapters one through nine and Appendix C.

Reference Books:

1.Kurose J F & Ross K.W: Computer Networking (Pearson)

2.Tanenbaum A S: Computer Networks (PHI) 4th Ed.

(DMSIT 02)

M.Sc.(Previous) DEGREE EXAMINATION, DECEMBER 2010.

First Year

Information Technology

Paper II — COMPUTER NETWORKS

Time : Three hours **Maximum : 75 marks**

SECTION A — (3 × 15 = 45 marks)

Answer any THREE of the following.

1. Explain various layers of OSI reference model.

2. Explain various multiple access protocols.

3. Explain the following:

(a) Circuit switching.

(b) Packet switching.

4. Discuss any three routing algorithms.

5. Describe DES algorithm and its significance in network security.

SECTION B — (5 × 5 = 25 marks)

Answer any FIVE of the following.

6. Distinguish between correction-oriented and correction-less communication.

7. Write short notes on LANs.

8. Write short notes on ATM.

9. Give any two strategies for congestion control.

10. What are the functions of IEEE 802.5 (token ring)?

11. What is Manchester encoding? Explain it with a neat diagram.

12. Write short notes on DNS.

13. Define cryptography. What are the purposes of it?

SECTION C — (5 × 1 = 5 marks)

Answer ALL of the following.

14. Define intranet.

15. What are correction-free protocols?

16. What are choke packets?

17. Define flooding?

18. What is a cipher?

(DMSIT 02)

M.Sc. DEGREE EXAMINATION, JUNE 2010.

Second Year

Information Technology

Paper II — COMPUTER NETWORKS

Time : Three hours **Maximum : 75 marks**

SECTION A — (3 x15 = 45 marks)

Answer any THREE of the following.

1. What is data communication? Explain its components in detail.
2. Explain different types of network topologies with a neat diagram.
3. Explain any three routing algorithm in networking.
4. Explain about the concept of switching in networks.
5. Explain the procedure involved in the calculation of IP address.

SECTION B — (5 x 5 = 25 marks)

Answer any FIVE of the following.

6. Distinguish between connection-oriented and connectionless communication network services.
7. Write in brief about TCP/IP reference model.
8. Write about Frequency-Division Multiple Access (FDMA) with a neat diagram.
9. Narrate any two network technologies.
10. Explain various services provided by the networks.
11. Explain the optimality principle of routing.
12. Write short notes on binary arithmetic.
13. Explain the RSA algorithm.

SECTION C — (5 x 1 = 5 marks)

Answer ALL of the following.

14. What is a network?
15. What is ALOHA?
16. Define packet switching.
17. What do you mean by congestion control?
18. Define message digest.

(DMSIT 02)

M.Sc. DEGREE EXAMINATION, MAY 2011.

First Year

INFORMATION TECHNOLOGY

Paper II — COMPUTER NETWORKS

Time : Three hours **Maximum : 75 marks**

SECTION A — (3x15 = 45 marks)

Answer any THREE of the following.

1. Discuss in brief different transmission media in respect of physical description, transmission characteristics and usages.
2. Explain about TCP/IP model with neat diagram.
3. How do you encode digital signal? Explain various methods.
4. Discuss the unicast routing procols in Network layer.
5. Explain about QOS and their techniques.

SECTION B — (5x5 = 25 marks)

Answer any FIVE of the following.

6. Explain Goback - N ARQ protocol with neat diagram.
7. What is DNS? What is the importance of it?
8. Explain about circuit switched networks.
9. What is channelization?
10. What is Virtual LAN?
11. Explain about IPV4.
12. Write about UDP protocol.
13. Explain about DES algorithm.

SECTION C — (5x1 = 5 marks)

Answer ALL questions.

14. What is a computer network?
15. What is Bluetooth?
16. Explain about TCP.
17. What is Ethernet?
18. What is cryptography?

UNIT – I

Structure

1.1 Introduction

Data communications and networking are changing the way we do business and the way we live. Business decisions have to be made ever more quickly, and the decision makers require immediate access to accurate information. Why wait a week for that report from Germany to arrive by mail when it could appear almost instantaneously through computer networks? Businesses today rely on computer networks and internetworks. But before we ask how quickly we can get hooked up, we need to know how networks operate, what types of technologies are available, and which design best fills which set of needs.

The development of the personal computer brought about tremendous changes for business, industry, science, and education. A similar revolution is occurring in data communications and networking. Technological advances are making it possible for communications links to carry more and faster signals. As a result, services are evolving to allow use of this expanded capacity. For example, established telephone services such as conference calling, call waiting, voice mail, and caller ID have been extended.

Research in data communications and networking has resulted in new technologies. One goal is to be able to exchange data such as text, audio, and video from all points in the world. We want to access the Internet to download and upload information quickly and accurately and at any time.

1.2 Data Communications

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. The term *telecommunication,* which includes telephony, telegraphy, and television, means communication at a distance (tele is Greek for "far").

The word data refers to information presented in whatever form is agreed upon by the parties creating and using the data.

Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable. For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs). The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. Delivery. The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.

2. Accuracy. The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.

3. Timeliness. The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery

means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called real-time transmission.

4. Jitter. Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 3ms. If some of the packets arrive with 3ms delay and others with 4ms delay, an uneven quality in the video is the result.

Components

A data communications system has five components (see Figure 1.1).

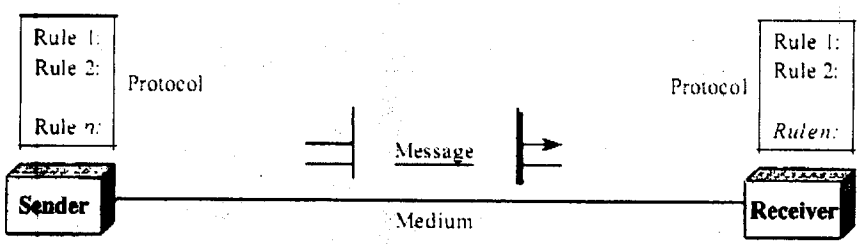


Figure 1.1 Five components of data communication

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 (0s or 1s). 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. Appendix A includes part of the Unicode.

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. Appendix B discusses several different numbering systems.

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. In the second case, there is a better representation of the image (better resolution), but more memory is needed to store the image.

After an image is divided into pixels, each pixel is assigned a bit pattern. The size and the value of the pattern depend on the image. For an image made of only black and white dots (e.g., a chessboard), l-bit pattern is enough to represent a pixel.

If an image is not made of pure white and pure black pixels, you can increase the size of the bit pattern to include gray scale. For example, to show four levels of gray scale, you can use 2-bit patterns. A black pixel can be represented by 00, a dark gray pixel by 01, a light gray pixel by 10, and a white pixel by 11.

There are several methods to represent color images. One method is called RGB, so called because each color is made of a combination of three primary colors: red, green, and blue. The intensity of each color is measured, and a bit pattern is assigned to it. Another method is called YCM, in which a color is made of a combination of three other primary colors: yellow, cyan, and magenta.

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. Again we can change video to a digital or an analog signal.

Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex as shown in Figure 1.2.

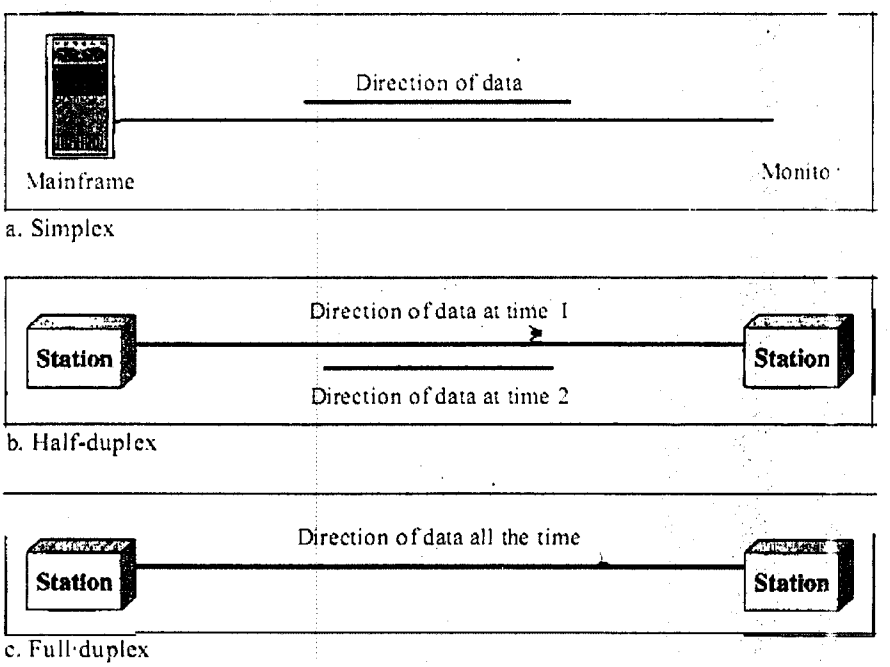


Figure 1.2 Data flow (simplex, half-duplex, and full-duplex)

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 (see Figure 1.2a).

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 (see Figure 1.2b).

The half-duplex mode is like a one-lane road with traffic allowed in both directions. When cars are traveling in one direction, cars going the other way must wait. 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.

The half-duplex mode is used in cases where there is no need for communication in both directions at the same time; the entire capacity of the channel can be utilized for each direction.

Full-Duplex

In full-duplex mode (also called duplex), both stations can transmit and receive simultaneously (see Figure 1.2c).

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. This sharing can occur in two ways: Either the link must contain two physically separate transmission paths, one for sending and the other for receiving; or the capacity of the channel is divided between signals traveling in both directions.

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.

1.3 Networks

A network is a set of devices (often referred to as nodes,) connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

Distributed Processing

Most networks use distributed processing, in which a task is divided among multiple computers. Instead of one single large machine being responsible for all aspects of a process, separate computer (usually a personal computer or workstation) handle a subset.

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

Performance 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. We often need more throughputs and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

Reliability In addition to accuracy of delivery, 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 and development, and implementing policies and procedures for recovery from breaches and data losses.

Physical Structures

Before discussing networks, we need to define some network attributes.

Type 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. For visualization purposes, it is simplest to imagine any link as a line drawn between two points. For communication to occur, two devices must be connected in some way to the same link at the same time. 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. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible (see Figure 1.3a). When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

Multipoint A multipoint (also called multidrop) connection is one in which more than two specific devices share a single link (see Figure 1.3b).

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.

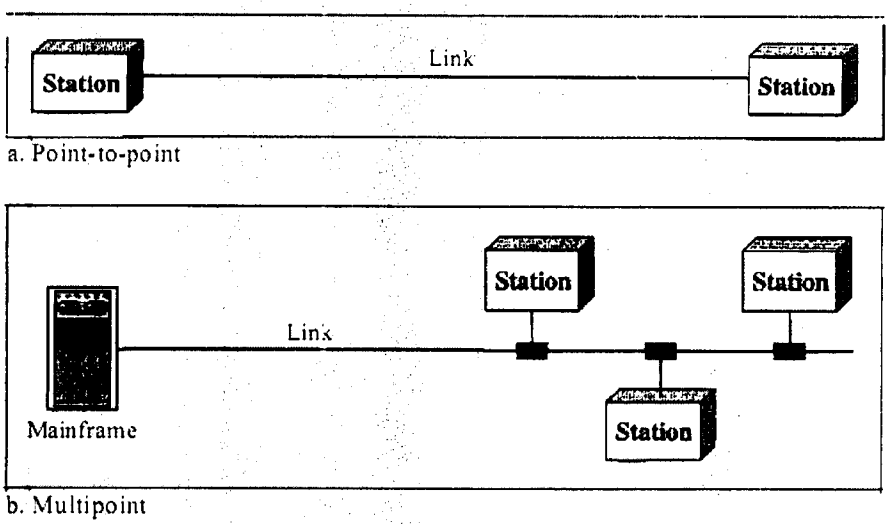


Figure 1.3 Type of Connection (a) Point-to-Point (b) Multipoint

Physical Topology

The term physical topology refers to the way in which a network is laid out physically: Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. There are four basic topologies possible: mesh, star, bus, and ring (see Figure 1.4).

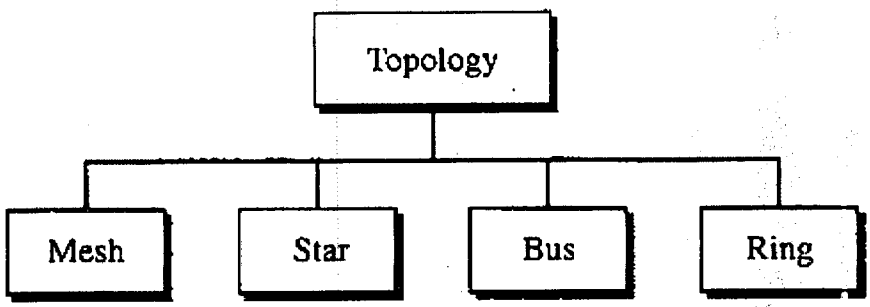


Figure 1.4 Categories of topology

Mesh

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. To find the number of physical links in a fully connected mesh network with ii nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to n - I nodes, node 2 must be connected to n - 1 nodes, and finally node n must be connected to n - 1 nodes. We need n (n - 1) physical links, however, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need n (n-1)/2 duplex-mode links.

To accommodate that many links, every device on the network must have n - 1 input/output (I/O) ports (see Figure 1.5) to be connected to the other n-1stations.

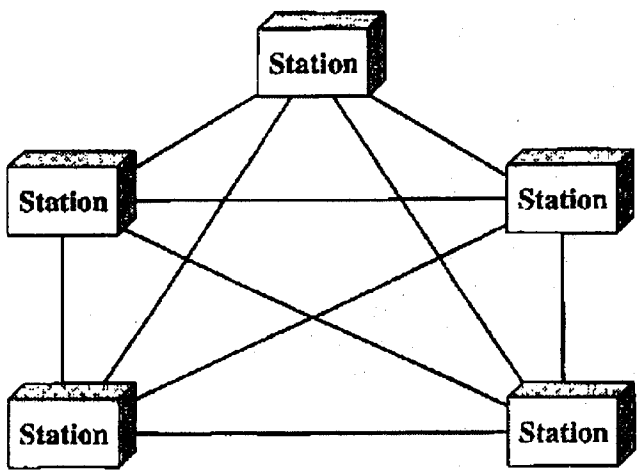


Figure 1.5 A fully connected mesh topology (five devices)

A mesh offers several advantages over other network topologies. First, 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. Second, a mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system. Third, 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. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate. Finally, the hardware required to connect each link (I/O ports and cable) can be prohibitively expensive. For these reasons a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies.

One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

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. Unlike a mesh topology, 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 (see Figure 1.6)

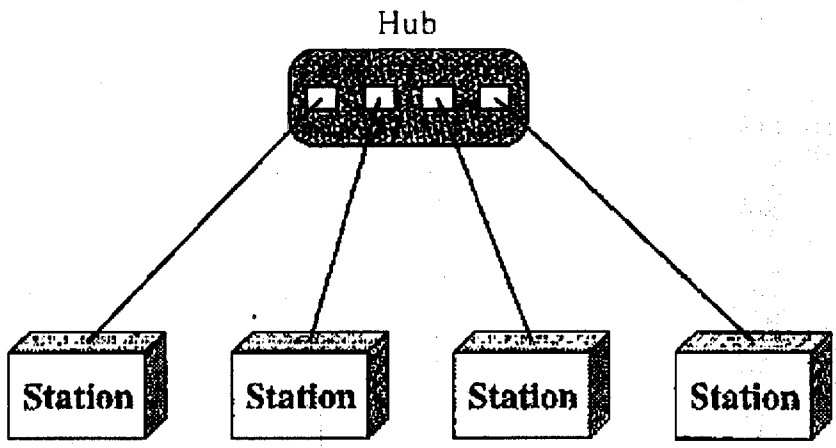


Figure 1.6 A star topology connecting four stations

A star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect it to any number of others. This factor also makes it easy to install and reconfigure. Far less cabling needs to be housed, and additions, moves, and deletions involve only one connection: between that device and the hub.

Other advantages include robustness. 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. As long as the hub is working, it can be used to monitor link problems and bypass defective links.

One big disadvantage of a 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.

Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is required in a star than in some other topologies (such as ring or bus).

The star topology is used in local-area networks (LANs), as high-speed LANs often use a star topology with a central hub.

Bus Topology

The preceding examples all describe point-to-point connections. A bus topology, on the other hand, is multipoint. One long cable acts as a backbone to link all the devices in a network (see Figure 1.7).

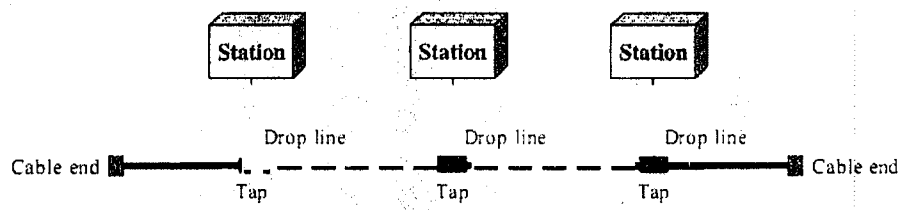


Figure 1.7 A bus topology connecting three stations

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. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, and then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the backbone.

Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected to a given length of cable. Adding new devices may therefore require modification or replacement of the backbone.

In addition, a 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.

Bus topology was the one of the first topologies used in the design of early local- area networks. Ethernet LANs can use a bus topology, but they are less popular.

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. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along (see Figure 1.8).

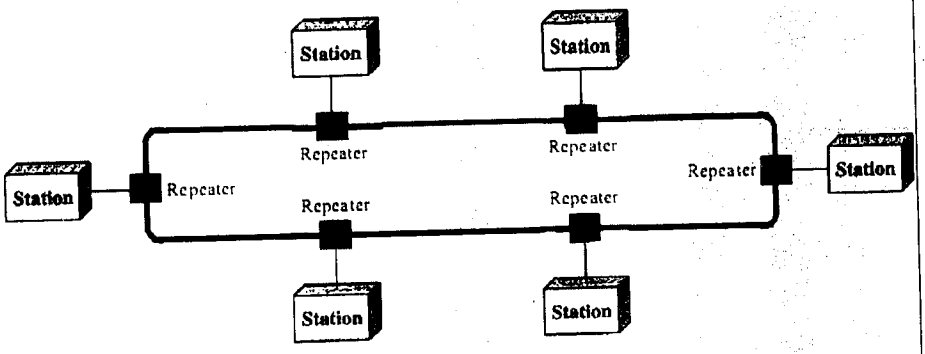


Figure 1.8 A ring topology connecting six stations

A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally in a ring, a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

Ring topology was prevalent when IBM introduced its local-area network Token Ring. Today, the need for higher-speed LANs has made this topology less popular.

Hybrid Topology

A network can be hybrid. For example, we can have a main star topology with each branch connecting several stations in a bus topology as shown in Figure 1.9.

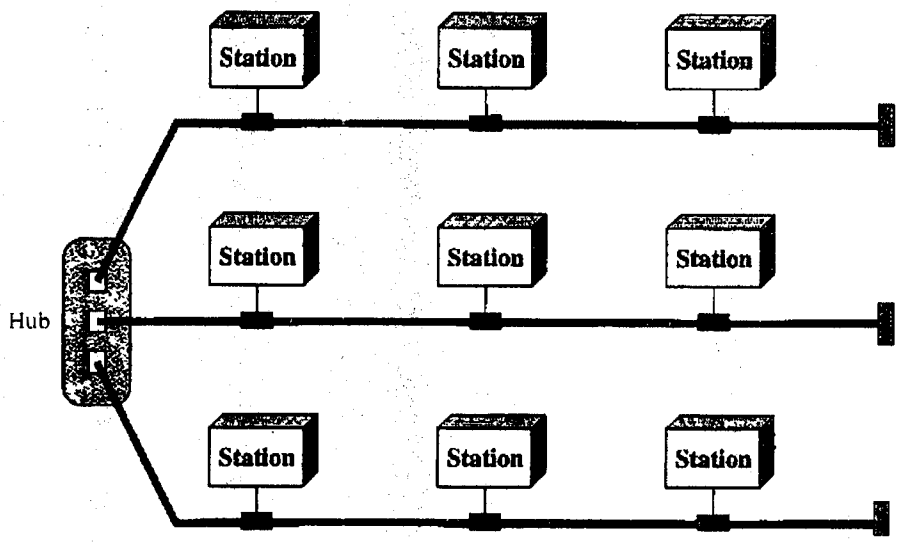


Figure 1.9 A hybrid topology: a star backbone with three bus networks

1.4 Signal

One of the major functions of the data communication is to move data in the form of electromagnetic signals across a transmission medium. Whether you are collecting numerical statistics from another computer, sending animated pictures from a design workstation, or causing a bell to ring at a distant control center, you are working with the transmission of data across network connections.

Generally, the data usable to a person or application are not in a form that can be transmitted over a network. For example, a photograph must first be changed to a form that transmission media can accept. Transmission media work by conducting energy along a physical path.

Analog and Digital

Both data and the signals that represent them can be either analog or digital in form.

Analog and Digital Data

Data can be analog or digital. The term analog data refers to information that is continuous; digital data refers to information that has discrete states. For example, an analog clock that has hour, minute, and second hands gives information in a continuous form; the movements of the hands are continuous. On the other hand, a digital clock that reports the hours and the minutes will change suddenly from 8:05 to 8:06.

Analog data, such as the sounds made by a human voice, take on continuous values. When someone speaks, an analog wave is created in the air. This can be captured by a microphone and converted to an analog signal or sampled and converted to a digital signal.

Digital data take on discrete values. For example, data are stored in computer memory in the form of 0s and 1s. They can be converted to a digital signal or modulated into an analog signal for transmission across a medium.

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 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 3.1 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 value.

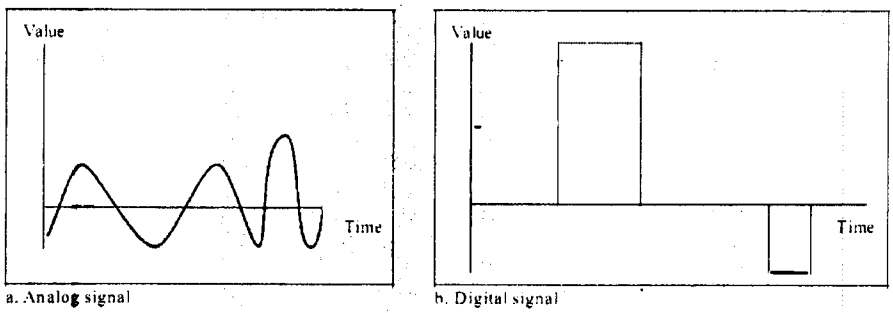


Figure 1.10 Comparison of analog and digital signals

Periodic and Nonperiodic Signals

Both analog and digital signals can take one of two forms: periodic or nonperiodic (sometimes refer to as aperiodic, because the prefix an in Greek means "non").

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 nonperiodic signal changes without exhibiting a pattern or cycle that repeats over time.

Both analog and digital signals can be periodic or nonperiodic. In data communications, we commonly use periodic analog signals (because they need less bandwidth) and nonperiodic digital signals (because they can represent variation in data).

Periodic Analog Signals

Periodic analog signals can be classified as simple or composite. A simple periodic analog signal, a sine wave, cannot be decomposed into simpler signals. A composite periodic analog signal is composed of multiple sine waves.

Sine Wave

The sine wave is the most fundamental form of a periodic analog signal. When we visualize it as a simple oscillating curve, its change over the course of a cycle is smooth and consistent, a continuous, rolling flow. Figure 1.11 shows a sine wave. Each cycle consists of a single arc above the time axis followed by a single arc below it.

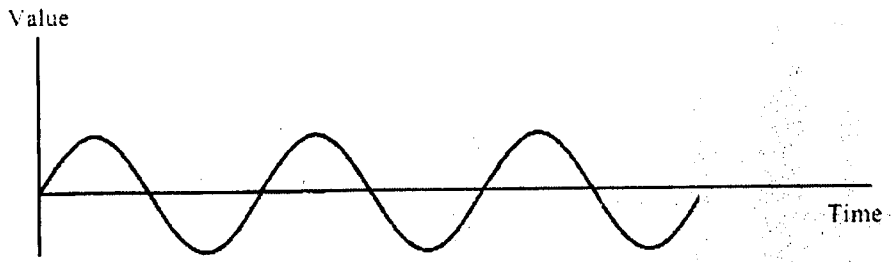
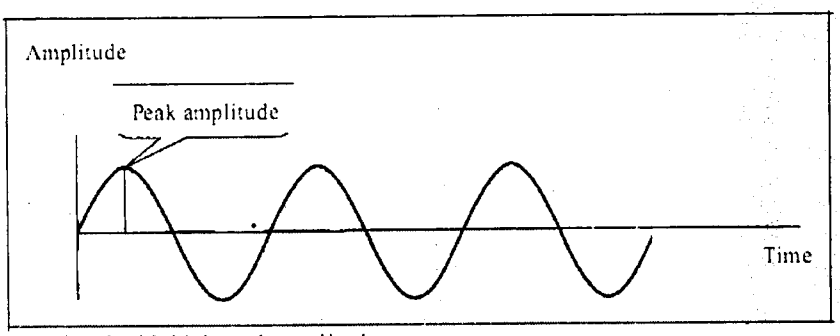


Figure 1.11 A sine wave

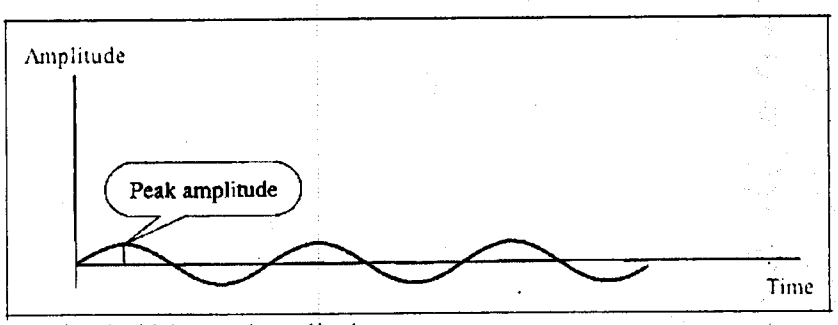
A sine wave can be represented by three parameters: the peak amplitude, the frequency, and the phase. These three parameters fully describe a sine wave.

Peak Amplitude

The peak amplitude of a signal is the absolute value of its highest intensity, proportional to the energy it carries. For electric signals, peak amplitude is normally measured in its. Figure 1.12 shows two signals and their peak amplitudes.



a. A signal with high peak amplitude



b. A signal with low peak amplitude

Figure 1.12 Two signals with the same phase and frequency, but different amplitudes

Period and Frequency

Period refers to the amount of time, in seconds, a signal needs to complete 1 cycle. Frequency refers to the number of periods in I s. Note that period and frequency are just one characteristic defined in two ways. Period is the inverse of frequency, and frequency is the inverse of period, as the following formulas show.

F = 1/T and T = 1/f

Period is formally expressed in seconds. Frequency is formally expressed in hertz (Hz), which is cycle per second. Units of period and frequency are shown in Table 1.1.

Unit	*Equivalent*	*Unit*	*Equivalent*
Seconds (s)	1 s	Hertz (Hz)	1 Hz
Milliseconds (ms)	10^{-3} s	Kilohertz (kHz)	10^{3} Hz
Microseconds (μs)	10-6 s	Megahertz (MHz)	10^{6} Hz
Nanoseconds (ns)	10^{-9} s	Gigahertz (GHz)	10^{9} Hz
Picoseconds (ps)	10^{-12} s	Terahertz (THz)	10^{12} Hz

Table 1.1 Units of period and frequency

Phase

The term phase describes the position of the waveform relative to time 0. If we think of the wave as something that can be shifted backward or forward along the time axis, phase describes the amount of that shift. It indicates the status of the first cycle.

Phase is measured in degrees or radians [360° is 2nrad; 1° is 2n/36Orad, and 1rad is 360/(2n)]. A phase shift of 360° corresponds to a shift of a complete period; a phase shift of 180° corresponds to a shift of one-half of a period; and a phase shift of 90° corresponds to a shift of one-quarter of a period (see Figure 1.13).

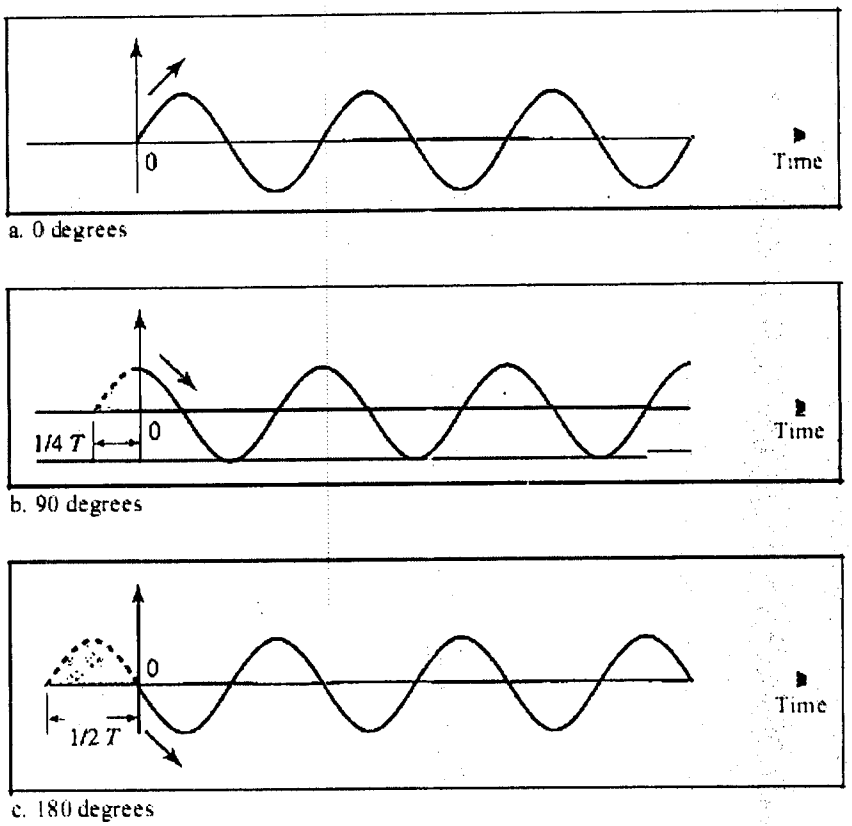


Figure 1.13 Three sine waves with the same amplitude and frequency, but different phases

Looking at Figure 1.13, we can say that

I. A sine wave with a phase of 00 starts at time 0 with zero amplitude. The amplitude is increasing.

2. A sine wave with a phase of 90° starts at time 0 with a peak amplitude. The amplitude is decreasing.

3. A sine wave with a phase of 180° starts at time 0 with a zero amplitude. The amplitude is decreasing.

Another way to look at the phase is in terms of shift or offset. We can say that

1. A sine wave with a phase of 0° is not shifted.

2. A sine wave with a phase of9O° is shifted to the left by cycle. However, note that the signal does not really exist before time 0.

3. A sine wave with a phase of 180° is shifted to the left by cycle. However, note that the signal does not really exist before time 0.

Wavelength

Wavelength is another characteristic of a signal traveling through a transmission medium. Wavelength binds the period or the frequency of a simple sine wave to the propagation speed of the medium (see Figure 1.14).

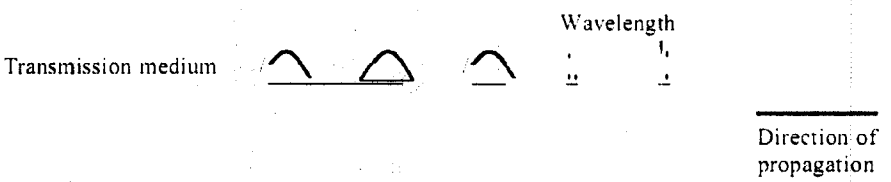


Figure 1.14 Wavelength and period

While the frequency of a signal is independent of the medium, the wavelength depends on both the frequency and the medium. Wavelength is a property of any type of signal. In data communications, we often use wavelength to describe the transmission of light in an optical fiber. The wavelength is the distance a simple signal can travel in one period.

Wavelength can be calculated if one is given the propagation speed (the speed of light) and the period of the signal. However, since period and frequency are related to each other, if we represent wavelength by λ propagation speed by c (speed of light), and frequency by f, we get

Wavelength = propagation speed x period = propagation speed / frequency

The propagation speed of electromagnetic signals depends on the medium and on the frequency of the signal. For example, in a vacuum, light is propagated with a speed of 3×10^8mls. That speed is lower in air and even lower in cable.

The wavelength is normally measured in micrometers (microns) instead of meters. For example, the wavelength of red light (frequency 4×10^{14}) in air is

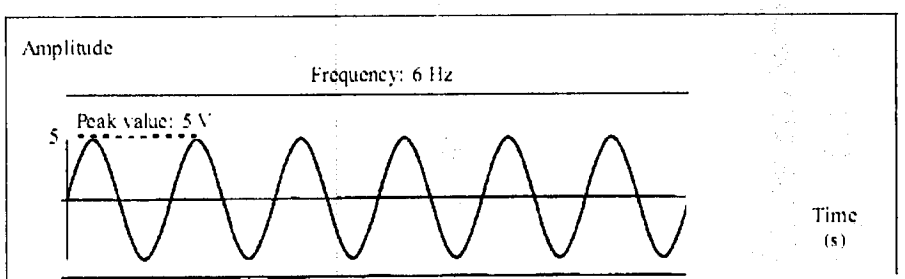
$\lambda = c/f = 3 \times 10^8 / 4 \times 10^{14} = 0.75 \times 10^{-6} = 0.75\mu m$

In a coaxial or fiber-optic cable, however, the wavelength is shorter (0.5 μm) because the propagation speed in the cable is decreased.

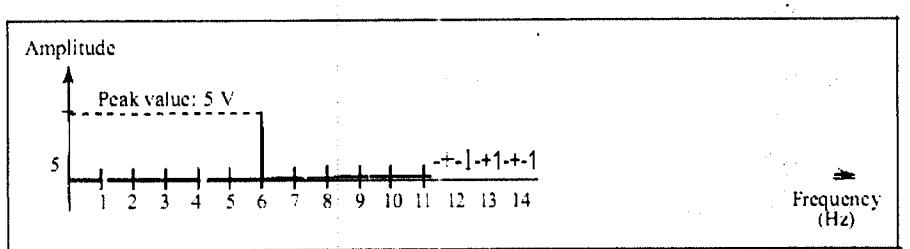
Time and Frequency Domains

A sine wave is comprehensively defined by its amplitude, frequency, and phase. We have been showing a sine wave by using what is called a time-domain plot. The time-domain plot shows changes in signal amplitude with respect to time (it is an amplitude-versus-time plot). Phase is not explicitly shown on a time-domain plot.

To show the relationship between amplitude and frequency, we can use what is called a frequency-domain plot. A frequency-domain plot is concerned with only the peak value and the frequency. Changes of amplitude during one period are not shown. Figure 1.15 shows a signal in both the time and frequency domains.



a. A sine wave in the time domain (**peak** value: 5 V, frequency: 6 Hz)



b. The same sine wave in the frequency domain (peak value: 5 V, frequency: 6 Hz)

Figure 1.15 The time-domain and frequency-domain plots of a sine wave

It is obvious that the frequency domain is easy to plot and conveys the information that one can find in a time domain plot. The advantage of the frequency domain is that we can immediately see the values of the frequency and peak amplitude. A complete sine wave is represented by

one spike. The position of the spike shows the frequency; its height shows the peak amplitude.

Composite Signals

So far, we have focused on simple sine waves. Simple sine waves have many applications in daily life. We can send a single sine wave to carry electric energy from one place to another. For example, the power company sends a single sine wave with a frequency of 60 Hz to distribute electric energy to houses and businesses. As another example, we can use a single sine wave to send an alarm to a security center when a burglar opens a door or window in the house. In the first case, the sine wave is carrying energy; in the second, the sine wave is a signal of danger.

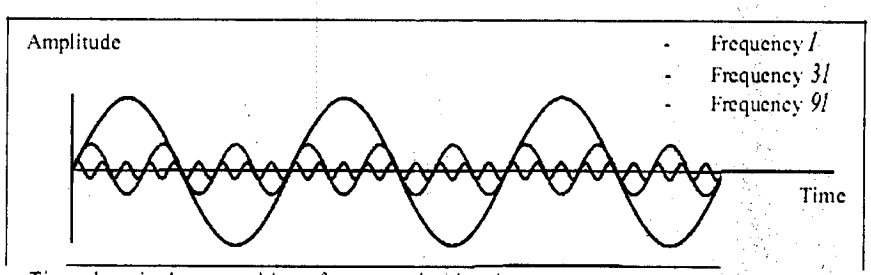
A composite signal can be periodic or nonperiodic. A periodic composite signal can be decomposed into a series of simple sine waves with discrete frequencies that have integer values (1, 2, 3, and so on). A nonperiodic composite signal can be decomposed into a combination of an infinite number of simple sine waves with continuous frequencies.

It is very difficult to manually decompose this signal into a series of simple sine waves. However, there are tools, both hardware and software, that can help us do the job. We are not concerned about how it is done; we are only interested in the result. Figure 1.16 shows the result of decomposing the above signal in both the time and frequency domains.

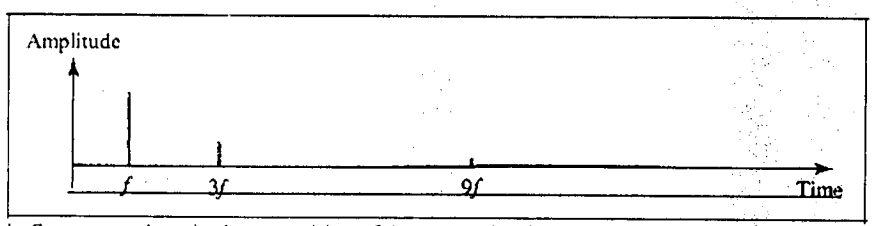
The amplitude of the sine wave with frequency f is almost the same as the peak amplitude of the composite signal. The amplitude of the sine wave with frequency 3fis one-third of that of the first, and the amplitude of the sine wave with frequency 9f is one-ninth of the first. The frequency of the sine wave with frequency f is the same as the frequency of the composite signal; it is called the fundamental frequency, or first harmonic. The sine wave with frequency 3f has a frequency of 3 times the fundamental frequency; it is called the third harmonic. The third sine wave with frequency 9f has a frequency of 9 times the fundamental frequency: it is called the ninth harmonic.

Note that the frequency decomposition of the signal is discrete it has frequencies f, 3f and 9f. Because f is an integral number, 3f and 9f are also integral numbers. There are no frequencies such as 1.2 for 2.6f. The

frequency domain of a periodic composite signal is always made of discrete spikes.



a. Time-domain decomposition of a composite signal



b. Frequency-domain decomposition of the composite signal

Figure 1.16 Decomposition of a composite periodic signal in the time and frequency domains

Bandwidth

The range of frequencies contained in a composite signal is its bandwidth. The bandwidth is normally a difference between two numbers. For example, if a composite signal contains frequencies between 1000 and 5000, its bandwidth is 5000- 1000, or 4000.

Figure 1.17 shows the concept of bandwidth. The figure depicts two composite signals, one periodic and the other nonperiodic. The bandwidth of the periodic signal contains all integer frequencies between 1000 and 5000(1000, 1001, 1002 ...). The bandwidth of the nonperiodic signals has the same range, but the frequencies are continuous.

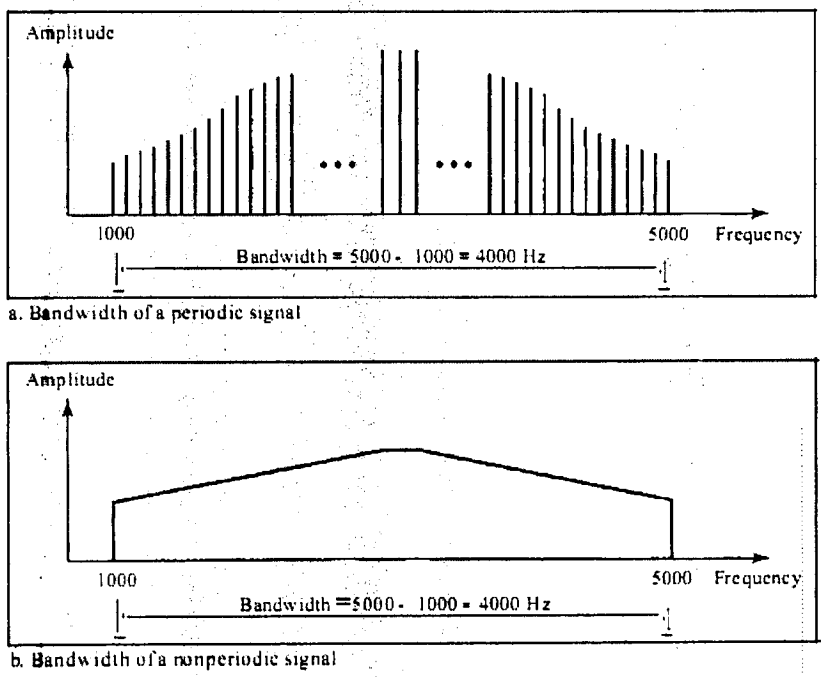
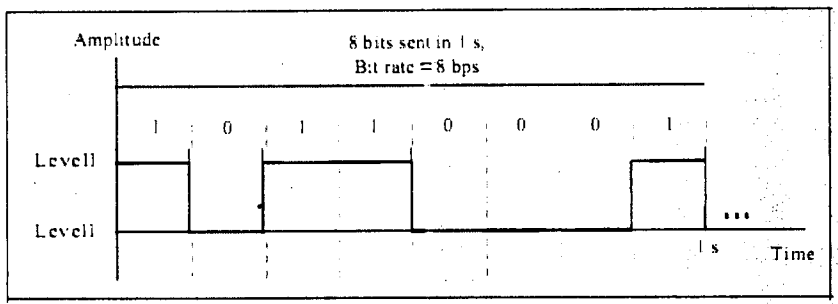


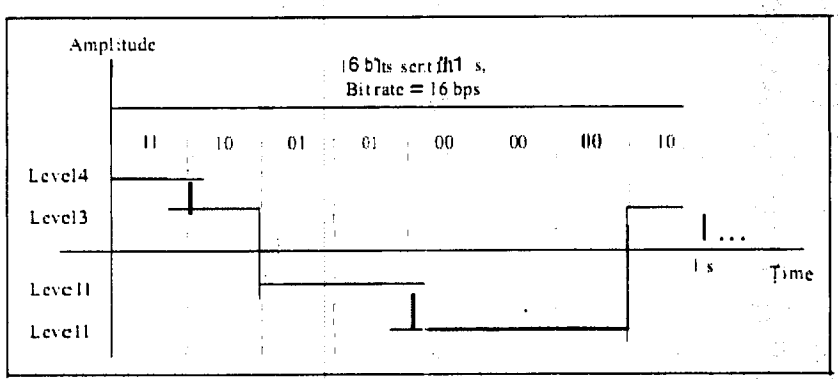
Figure 1.17 The bandwidth of periodic and nonperiodic composite signals

Digital Signals

In addition to being represented by an analog signal, information can also be represented by a digital signal. For example, a 1 can be encoded as a positive voltage and a 0 as zero voltage. A digital signal can have more than two levels. In this case, we can send more than 1 bit for each level. Figure 1.18 shows two signals, one with two levels and the other with four.



a. A digital signal with two levels



b. A digital signal with four levels

Figure 1.18 Two digital signals: one with two signal levels and the other with four signal levels

Bit Rate

Most digital signals arc nonperiodic and thus period and frequency arc not appropriate characteristics. Another term -bit rate (instead of frequency)-is used to describe digital signals. The bit rate is the number of bits sent in one second, expressed in bits per second (bps). Figure 1.18 shows the bit rate for two signals.

Bit Length

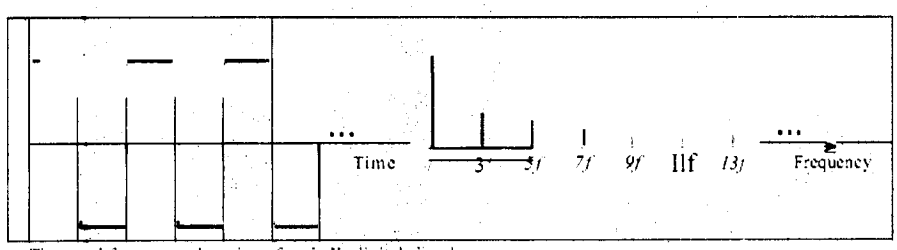
We discussed the concept of the wavelength for an analog signal: the distance one cycle occupies on the transmission medium. We can define something similar for a digital signal: the bit length. The bit length is the distance one bit occupies on the transmission medium.

Bit length = propagation speed x bit duration

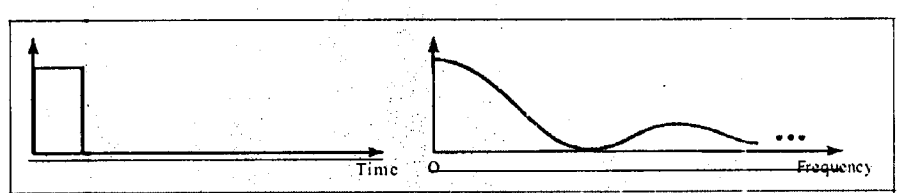
Digital Signal as a Composite Analog Signal

Based on Fourier analysis, a digital signal is a composite analog signal. The bandwidth is infinite, as you may have guessed. We can intuitively come up with this concept when we consider a digital signal. A digital signal, in the time domain, comprises connected vertical and horizontal line segments. A vertical line in the time domain means a frequency of infinity (sudden change in time); a horizontal line in the time domain means a frequency of zero (no change in time). Going from a frequency of zero to a frequency of infinity (and vice versa) implies all frequencies in between are part of the domain.

Fourier analysis can be used to decompose a digital signal. If the digital signal is periodic, which is rare in data communications, the decomposed signal has a frequency domain representation with an infinite bandwidth and discrete frequencies. If the digital signal is nonperiodic, the decomposed signal still has an infinite bandwidth, but the frequencies are continuous. Figure 1.19 shows a periodic and a nonperiodic digital signal and their bandwidths.



a. Time and frequency domains of **periodic** digital signal



b. Time and frequency domains of nonperiodic digital signal

Figure 1.19 The time and frequency domains of periodic and nonperiodic digital signals

Note that both bandwidths are infinite, but the periodic signal has discrete frequencies while the nonperiodic signal has continuous frequencies.

Transmission of Digital Signals

The previous discussion asserts that a digital signal, periodic or nonperiodic, is a composite analog signal with frequencies between zero

and infinity. For the remainder of the discussion, let us consider the case of a nonperiodic digital signal, similar to the ones we encounter in data communications. The fundamental question is how can we send a digital signal from point A to point B? We can transmit a digital signal by using one of two different approaches: baseband transmission or broadband transmission (using modulation).

Baseband Transmission

Baseband transmission means sending a digital signal over a channel without changing the digital signal to an analog signal. Figure 1.20 shows baseband transmission.

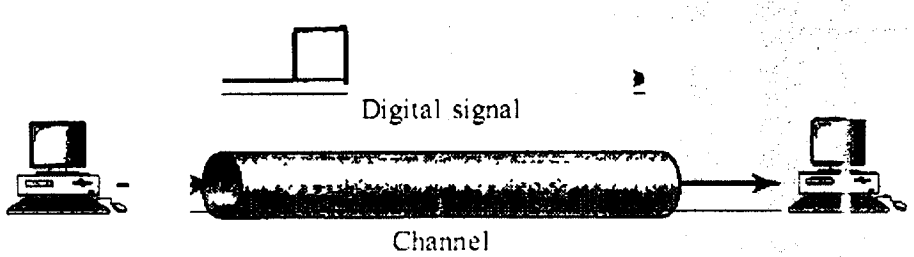


Figure 1.20 Baseband transmission

Baseband transmission requires that we have a low-pass channel, a channel with a bandwidth that starts from zero. This is the case if we have a dedicated medium with a bandwidth constituting only one channel. For example, the entire bandwidth of a cable connecting two computers is one single channel. As another example, we may connect several computers to a bus, but not allow more than two stations to communicate at a time. Again we have a low-pass channel, and we can use it for baseband communication.

Distortion

Distortion means that the signal changes its form of shape. Distortion can occur in a composite signal made of different frequencies. Each signal component has its own propagation speed (sec the next section) 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. In other words, signal components at the receiver have phases different from what they had at the sender. The

shape of the composite signal is therefore not the same. Figure 1.21 shows the effect of distortion on a composite signal.

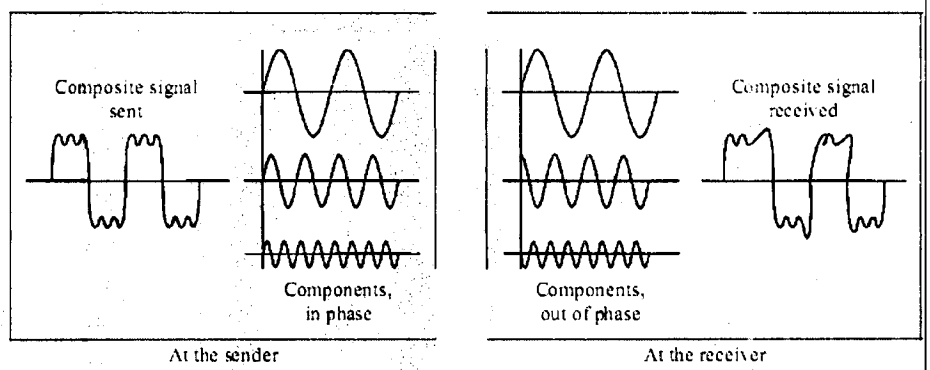


Figure 1.21 Distortion

Noise

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. Figure 1.22 shows the effect of noise on a signal.

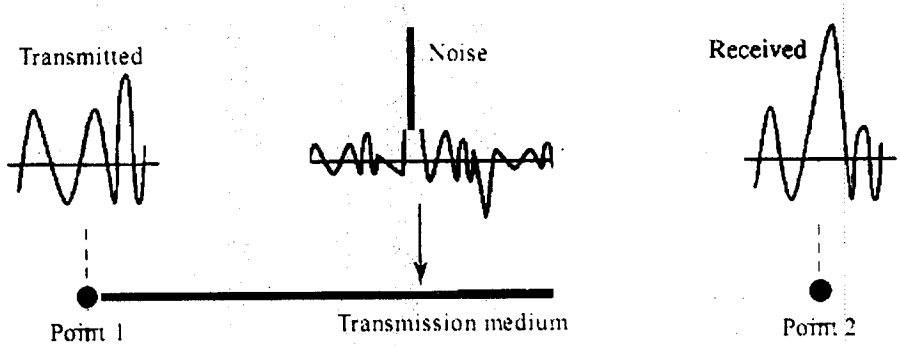


Figure 1.22Noise

Signal-to-Noise Ratio (SNR)

As we will see later, to find the theoretical bit rate limit, we need to know the ratio of the signal power to the noise power. The signal-to-noise ratio is defined as

SNR = average signal power / average noise power

We need to consider the average signal power and the average noise power because these may change with time. Figure 1.23 shows the idea of SNR.

SNR is actually the ratio of what is wanted (signal) to what is not wanted (noise). A high SNR means the signal is less corrupted by noise; a low SNR means the signal is more corrupted by noise.

Because SNR is the ratio of two powers, it is often described in decibel units, SNR_{dB}, defined as

$$SNR_{dB} = 10\log_{10} SNR$$

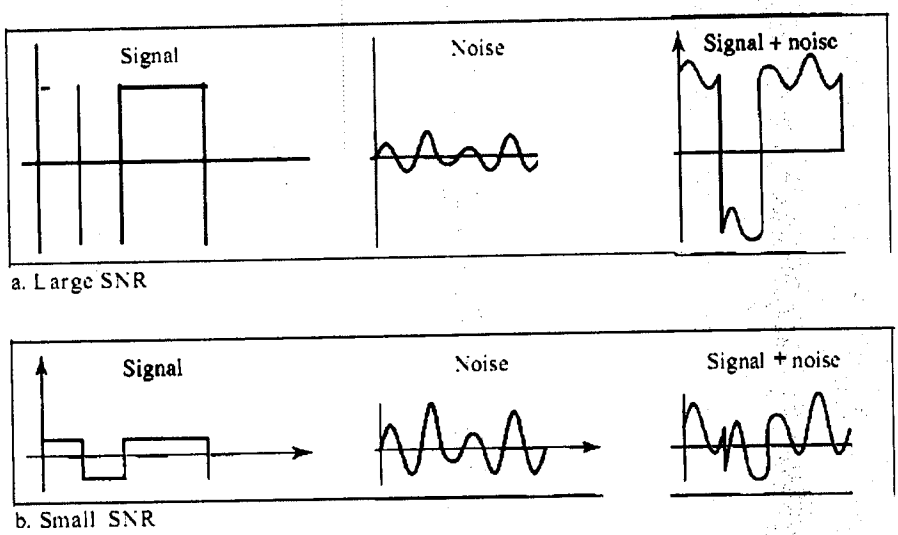


Figure 1.23 Two cases of SNR: a high SNR and a low SNR

1.5 Encoding and Decoding

A computer network is designed to send information from one point to another. This information needs to be converted to either a digital signal or an analog signal for transmission.

We discussed the advantages and disadvantages of digital transmission over analog transmission. In this section, we show the schemes and techniques that we use to transmit data digitally. First, we discuss digital-to-digital conversion techniques, methods which convert digital data to digital signals. Second, we discuss analog-to-digital conversion techniques, methods which change an analog signal to a digital signal finally, we discuss transmission modes.

Digital-to-Digital Conversion

The data can be either digital or analog. We also said that signals that represent data can also be digital or analog. In this section, we see how we can represent digital data by using digital signals. The conversion involves three techniques: line coding, block coding, and scrambling. Line coding is always needed; block coding and scrambling mayor may not be needed.

Line Coding

Line coding is the process of converting digital data to digital signals. We assume that data, in the form of text, numbers, graphical images, audio, or video, are stored in computer memory as sequences of bits. Line coding converts a sequence of bits to a digital signal. At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are recreated by decoding the digital signal. Figure 1.24 shows the process.

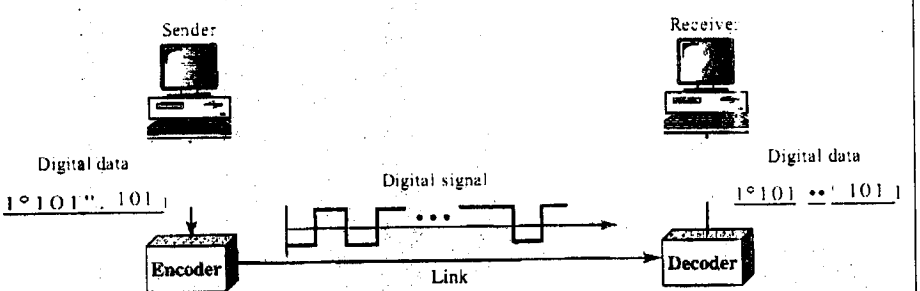


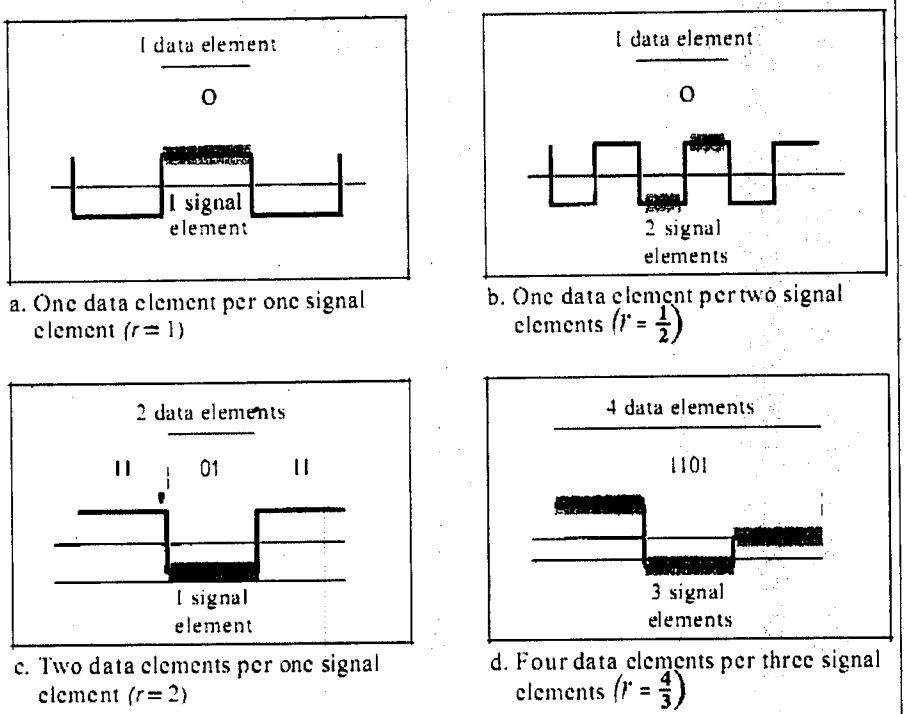
Figure 1.24 Line coding and decoding

Characteristics

Before discussing different line coding schemes, we address their common characteristics.

Signal Element Versus Data Element Let us distinguish between a data element and a signal element. In data communications, our goal is to send data elements. A data element is the smallest entity that can represent a piece of information: this is the bit. In digital data communications, a signal element carries data elements. A signal element is the shortest unit (time wise) of a digital signal. In other words, data elements are what we need to send; signal elements are what we can send. Data elements are being carried; signal elements are the carriers.

We define a ratio r which is the number of data elements carried by each signal element. Figure 1.25 shows several situations with different values of r.



a. One data element per one signal element ($r = 1$)

b. One data element per two signal elements ($r = \frac{1}{2}$)

c. Two data elements per one signal element ($r = 2$)

d. Four data elements per three signal elements ($r = \frac{4}{3}$)

Figure 1.25 Signal element versus data element

In part a of the figure, one data element is carried by one signal element (r = 1). In part b of the figure, we need two signal elements (two transitions) to carry each data element (r =1/2). We will see later that the extra signal element is needed to guarantee synchronization. In part c of the figure, a signal element carries two data elements (r = 2). Finally, in part d, a group of4 bits is being carried by a group of three signal elements (r = 4/3). For every line coding scheme we discuss, we will give the value of r.

An analogy may help here. Suppose each data element is a person who needs to be carried from one place to another. We can think of a signal element as a vehicle that can carry people. When r 1, it means each person is driving a vehicle. When r > 1, it means more than one person is travelling in a vehicle (a carpool, for example). We can also have the case where one person is driving a car and a trailer (r = 1/2).

Data Rate Versus Signal Rate The data rate defines the number of data elements (bits) sent in Is. The unit is bits per second (bps). The signal rate is the number of signal elements sent in Is. The unit is the baud. There are several common terminologies used in the literature. The data rate is sometimes called the bit rate; the signal rate is sometimes called the pulse rate, the modulation rate, or the baud rate.

One goal in data communications is to increase the data rate while decreasing the signal rate. Increasing the data rate increases the speed of transmission; decreasing the signal rate decreases the bandwidth requirement. In our vehicle-people analogy, we need to carry more people in fewer vehicles to prevent traffic jams. We have a limited bandwidth in our transportation system.

We now need to consider the relationship between data rate and signal rate (bit rate and baud rate). This relationship, of course, depends on the value of r. It also depends on the data pattern. If we have a data pattern of all is or all 0s, the signal rate may be different from a data pattern of alternating 0s and 1s. To derive a formula for the relationship, we need to define three cases: the worst, best, and average. The worst case is when we need the maximum signal rate; the best case is when we need the minimum. In data communications, we are usually interested in the average case. We can formulate the relationship between data rate and signal rate as

$$S = c \times N \times 1/r \text{ baud}$$

where N is the data rate (bps); c is the case factor, which varies for each case; S is the number of signal elements; and r is the previously defined factor.

Bandwidth is a digital signal that carries information is nonperiodic. We also showed that the bandwidth of a nonperiodic signal is continuous with an infinite range. However, most digital signals we encounter in real life have a

bandwidth with finite values. In other words, the bandwidth is theoretically infinite, but many of the components have such small amplitude that they can be ignored. The effective bandwidth is finite. From now on, when we talk about the bandwidth of a digital signal, we need to remember that we are talking about this effective bandwidth.

'We can say that the baud rate, not the bit rate, determines the required bandwidth for a digital signal. If we use the transportation analogy, the number of vehicles affects the traffic, not the number of people being carried. More changes in the signal mean injecting more frequencies into the signal. (Recall that frequency means change and change means frequency.) The bandwidth reflects the range of frequencies we need. There is a relationship between the baud rate (signal rate) and the bandwidth. Bandwidth is a complex idea. When we talk about the bandwidth, we normally define a range of frequencies. We need to know where this range is located as well as the values of the lowest and the highest frequencies. In addition, the amplitude (if not the phase) of each component is an important issue. In other words, we need more information about the bandwidth than just its value; we need a diagram of the bandwidth. We will show the bandwidth for most schemes we discuss in the chapter. For the moment, we can say that the bandwidth (range of frequencies) is proportional to the signal rate (baud rate). The minimum bandwidth can be given as

$B_{min} = c \times N \times 1/r$

We can solve for the maximum data rate if the bandwidth of the channel is given.

$N_{max} = 1/c \times B \times r$

Baseline Wandering In decoding a digital signal, the receiver calculates a running average of the received signal power. This average is called the baseline. The incoming signal power is evaluated against this baseline to determine the value of the data element. A long string of 0s or 1s can cause a drift in the baseline (baseline wandering) and make it difficult for the receiver to decode correctly. A good line coding scheme needs to prevent baseline wandering.

DC Components When the voltage level in a digital signal is constant for a while, the spectrum creates very low frequencies (results of Fourier

analysis). These frequencies around zero, called DC (direct-current) components, present problems for a system that cannot pass low frequencies or a system that uses electrical coupling (via a transformer). For example, a telephone line cannot pass frequencies below 200 Hz. Also a long-distance link may use one or more transformers to isolate different parts of the line electrically. For these systems, we need a scheme with no DC component.

Self-synchronization To correctly interpret the signals received from the sender, the receiver's bit intervals must correspond exactly to the sender's bit intervals. If the receiver clock is faster or slower, the bit intervals are not matched and the receiver might misinterpret the signals.

A self-synchronizing digital signal includes timing information in the data being transmitted. This can be achieved if there are transitions in the signal that alert the receiver to the beginning, middle, or end of the pulse. If the receiver's clock is out of synchronization, these points can reset the clock.

Built-in Error Detection It is desirable to have a built-in error-detecting capability in the generated code to detect some of or all the errors that occurred during transmission. Some encoding schemes that we will discuss have this capability to some extent.

Immunity to Noise and Interference Another desirable code characteristic is a code that is immune to noise and other interferences. Some encoding schemes that we will discuss have this capability.

Complexity A complex scheme is more costly to implement than a simple one. For example, a scheme that uses four signal levels is more difficult to interpret than one that uses only two levels.

Line Coding Schemes

We can roughly divide line coding schemes into five broad categories, as shown in Figure 1.26.

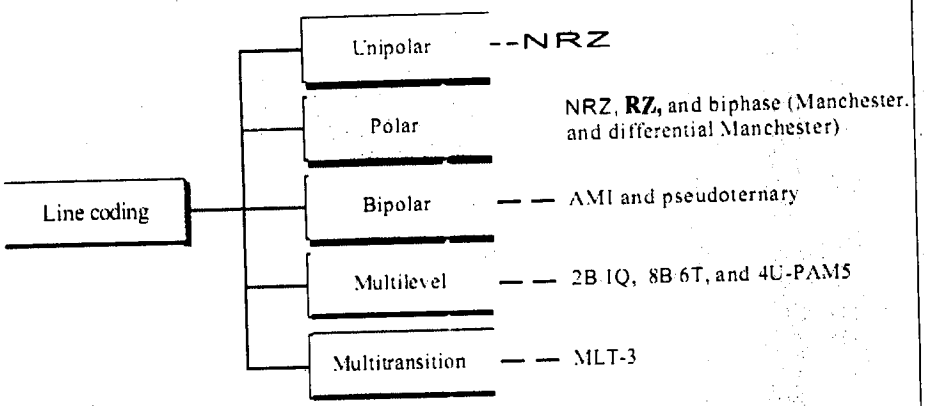


Figure 1.26 Line coding schemes

There are several schemes in each category. We need to be familiar with all schemes discussed in this section to understand the rest of the book. This section can be used as a reference for schemes encountered later.

Unipolar Scheme

In a unipolar scheme, all the signal levels are on one side of the time axis, either above or below.

NRZ (Non-Return-to-Zero) Traditionally, a unipolar scheme was designed as a non-return-to-zero (NRZ) scheme in which the positive voltage defines bit I and the zero voltage defines bit 0. It is called NRZ because the signal does not return to zero at the middle of the bit. Figure 1.27 shows a unipolar NRZ scheme.

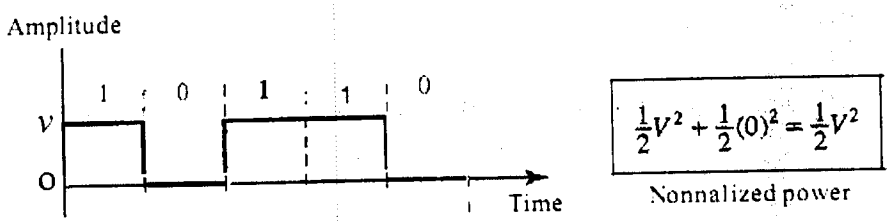


Figure 1.27 Unipolar NRZ scheme

Compared with its polar counterpart (see the next section), this scheme is very costly. As we will see shortly, the normalized power (power needed to send 1 bit per unit line resistance) is double that for polar NRZ. For this reason, this scheme is normally not used in data communications today.

Polar Schemes

In polar schemes, the voltages are on the both sides of the time axis. For example, the voltage level for 0 can be positive and the voltage level for 1, can be negative.

Non-Return-to-Zero (NRZ) In polar NRZ encoding, we use two levels of voltage amplitude. We can have two versions of polar NRZ: NRZ-L and NRZ-I, as shown in Figure 1.28. The figure also shows the value of i; the average baud rate, and the bandwidth. In the first variation, NRZ-L (NRZ-Level), the level of the voltage determines the value of the bit. In the second variation, NRZ-I (NRZ-Invert), the change or lack of change in the level of the voltage determines the value of the bit. If there is no change, the bit is 0; if there is a change, the bit is 1.

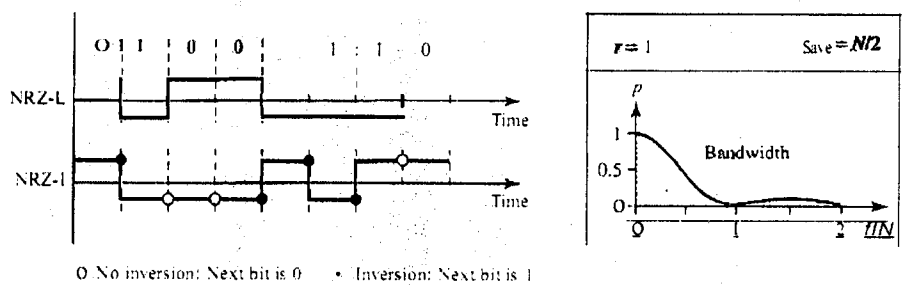


Figure 1.28 Polar NRZ-L and NRZ-I schemes

Let us compare these two schemes based on the criteria we previously defined. Although baseline wandering is a problem for both variations, it is twice as severe in NRZ-L. If there is a long sequence of 0s or 1s in NRZ-L, the average signal power becomes skewed. The receiver might have difficulty discerning the bit value. In NRZ-I this problem occurs only for a long sequence of 0s. If somehow we can eliminate the long sequence of 0s, we can avoid baseline wandering. We will see shortly how this can be done.

The synchronization problem (sender and receiver clocks are not synchronized) also exists in both schemes. Again, this problem is more serious in NRZ-L than in NRZ-I. While a long sequence of as can cause a problem in both schemes, a long sequence of is affects only NRZ-L.

Another problem with NRZ-L occurs when there is a sudden change of polarity in the system. For example, if twisted-pair cable is the medium, a

change in the polarity of the wire results in all as interpreted as 1s and all 1s interpreted as. NRZ-I does not have this problem. Both schemes have an average signal rate of N/2 Bd.

Let us discuss the bandwidth. Figure 1.28 also shows the normalized bandwidth for both variations. The vertical axis shows the power density (the power for each 1Hz of bandwidth); the horizontal axis shows the frequency. The bandwidth reveals a very serious problem for this type of encoding. The value of the power density is very high around frequencies close to zero. This means that there are DC components that carry a high level of energy.

Return to Zero (RZ) The main problem with NRZ encoding occurs when the sender and receiver clocks are not synchronized. The receiver does not know when one bit has ended and the next bit is starting. One solution is the return-to-zero (RZ) scheme, which uses three values: positive, negative, and zero. In RZ, the signal changes not between bits but during the bit. In Figure 1.29 we see that the signal goes to 0 in the middle of each bit. It remains there until the beginning of the next bit. The main disadvantage of RZ encoding is that it requires two signal changes to encode a bit and therefore occupies greater bandwidth. The same problem we mentioned, a sudden change of polarity resulting in all as interpreted as Is and all is interpreted as, still exist here, but there is no DC component problem. Another problem is the complexity: RZ uses three levels of voltage, which is more complex to create and discern. As a result of all these deficiencies, the scheme is not used today. Instead, it has been replaced by the better-performing Manchester and differential Manchester schemes (discussed next).

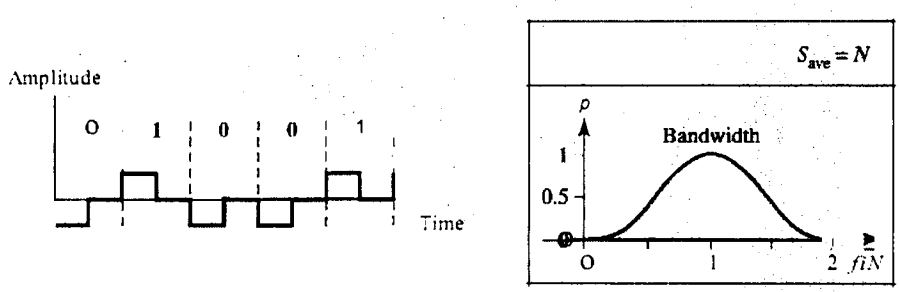


Figure 1.29 Polar RZ scheme

Biphase: Manchester and Differential Manchester The idea of RZ (transition at the middle of the bit) and the idea of NRZ-L are combined into

the Manchester scheme. In Manchester encoding, the duration of the bit is divided into two halves. The voltage remains at one level during the first half and moves to the other level in the second half. The transition at the middle o f the bit provides synchronization. Differential Manchester, on the other hand, combines the ideas of RZ and NRZ-I. There is always a transition at the middle of the bit. But the bit values arc determined at the beginning of the bit. If the next bit is 0, there is a transition; if the next bit is 1, there is none. Figure 1.30 shows both Manchester and differential Manchester encoding.

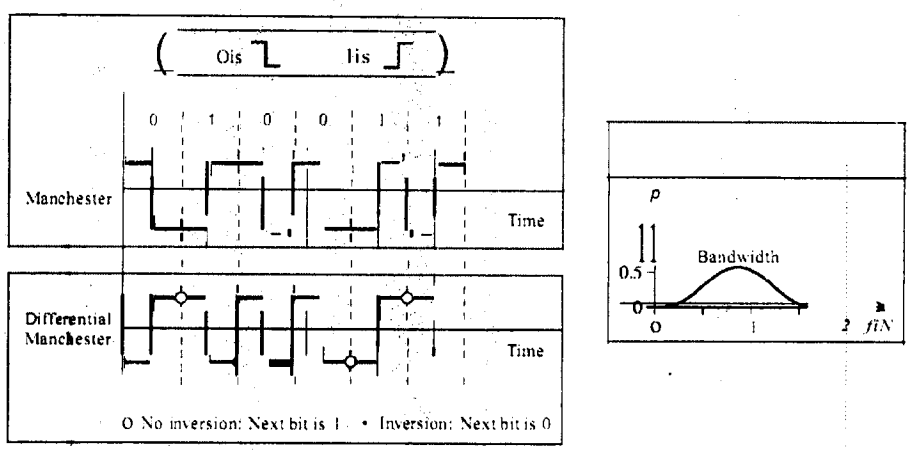


Figure 1.30 Polar biphase: Manchester and differential Manchester schemes

The Manchester scheme overcomes several problems associated with NRZ-L, and differential Manchester overcomes several problems associated with NRZ-I. First, there is no baseline wandering. There is no DC component because each bit has a positive and negative voltage contribution. The only drawback is the signal rate. The signal rate for Manchester and differential Manchester is double that for NRZ. The mason is that there is always one transition at the middle of the bit and maybe one transition at the end of each bit. Figure 1.30 shows both Manchester and differential Manchester encoding schemes. Note that Manchester and differential Manchester schemes are also called biphase schemes.

Bipolar Schemes

In bipolar encoding (sometimes called multilevel binary, there are three voltage levels: positive, negative, and zero. The voltage level for one data

element is at zero, while the voltage level for the other element alternates between positive and negative.

AMI and pseudo ternary Figure 1.31 shows two variations of bipolar encoding: AMI and pseudo ternary. A common bipolar encoding scheme is called bipolar alternate mark inversion (AMI). In the term alternate mark inversion, the word mark comes from telegraphy and means 1. So AMI means alternate 1 inversion. A neutral zero voltage represents binary 0. Binary 1s are represented by alternating positive and negative voltages. A variation of AMI encoding is called pseudo ternary in which the 1 bit is encoded as a zero voltage and the 0 bit is encoded as alternating positive and negative voltages.

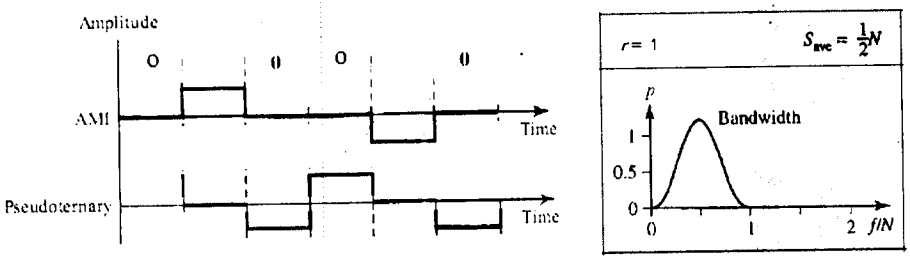


Figure 1.31 Bipolar schemes: AMI and pseudo ternary

The bipolar scheme was developed as an alternative to NRZ. The bipolar scheme has the same signal rate as NRZ, but there is no DC component. The NRZ scheme has most of its energy concentrated near zero frequency, which makes it unsuitable for transmission over channels with poor performance around this frequency. The concentration of the energy in bipolar encoding is around frequency N/2. Figure 1.31 shows the typical energy concentration for a bipolar scheme.

8B6T

A very interesting scheme is eight binary, six ternary (8B6T). This code is used with 100BASE-4T cable. The idea is to encode a pattern of 8 bits as a pattern of 6 signal elements, where the signal has three levels (ternary). In this type of scheme, we can have $2^8 = 256$ different data patterns and $3^6 = 478$ different signal patterns. The mapping table is shown in Appendix D. There are 478 – 256 = 222 redundant signal elements that provide synchronization and error detection. Part of the redundancy is also used to provide DC balance. Each signal pattern has a weight of 0 or +1 DC values. This means that there is no pattern with the weight -1. To make the

whole stream DC balanced, the sender keeps track of the weight. If two groups of weight 1 are encountered one after another, the first one is sent as is, while the next one is totally inverted to give a weight of -1.

4D-PAMS

The last signaling scheme we discuss in this category is called four-dimensional five-level pulse amplitude modulation (4D-PAMS). The 4D means that data is sent over four wires at the same time. It uses five voltage levels, such as -2, -1, 0, 1, and 2. However, one level, level 0, is used only for forward error detection. If we assume that the code is just one-dimensional, the four levels create something similar to 8B4Q. In other words, an 8-bit word is translated to a signal element of four different levels. The worst signal rate for this imaginary one-dimensional version is N x 4/8, or N/2.

The technique is designed to send data over four channels (four wires). This means the signal rate can be reduced to N/8, a significant achievement. All 8 bits can be fed into a wire simultaneously and sent by using one signal element. The point here is that the four signal elements comprising one signal group are sent simultaneously in a four-dimensional setting. Figure 1.32 shows the imaginary one-dimensional and the actual four-dimensional implementation. Gigabit LANs use this technique to send 1-Gbps data over four copper cables that can handle 125Mbaud. This scheme has a lot of redundancy in the signal pattern because 2^8 data patterns are matched to $4^4 = 256$ signal patterns. The extra signal patterns can be used for other purposes such as error detection.

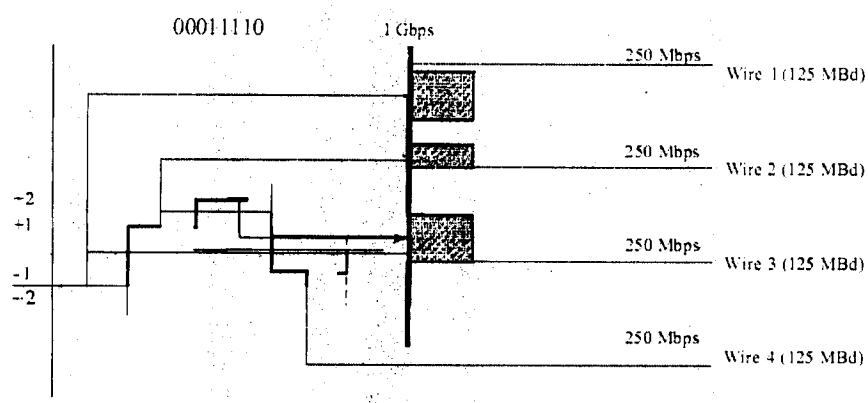


Figure 1.32 Multilevel: 4D-PAM5 scheme

Multiline Transmission: MLT-3

NRZ-I and differential Manchester are classified as differential encoding but use two transition rules to encode binary data (no inversion, inversion). If we have a signal with more than two levels, we can design a differential encoding scheme with more than two transition rules. MLT-3 is one of them. The multiline transmission, three level (MLT-3) scheme uses three levels (+V, 0, and - V) and three transition rules to move between the levels.

1. If the next bit is 0, there is no transition.

2. If the next bit is 1 and the current level is not 0, the next level is 0.

3. If the next bit is 1 and the current level is 0, the next level is the opposite of the last nonzero level.

The behavior of MLT-3 can best be described by the state diagram shown in Figure 1.33. The three voltage levels (-V, 0, and +V are shown by three states (ovals). The transition from one state (level) to another is shown by the connecting lines. Figure 1.33 also shows two examples of an MLT-3 signal.

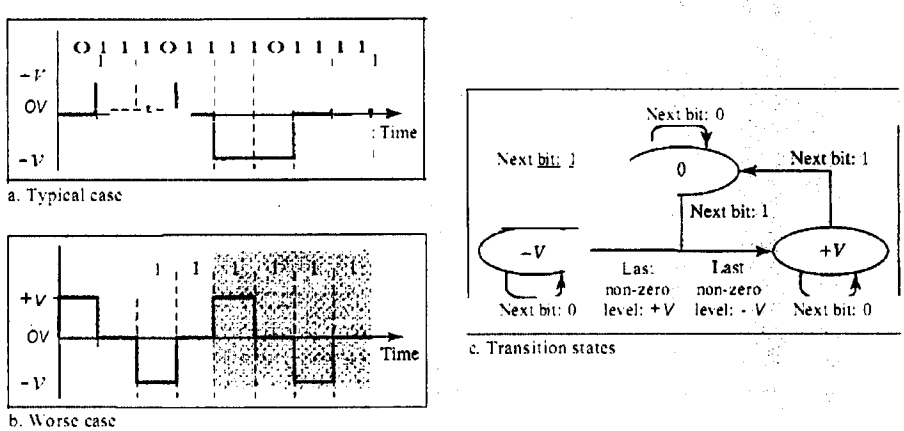


Figure 1.33 Multitransition: MLT-3 scheme

One might wonder why we need to use MLT-3, a scheme that maps one bit to one signal element. The signal rate is the same as that for NRZ-I, but with greater complexity (three levels and complex transition rules). It turns out that the shape of the signal in this scheme helps to reduce the required bandwidth. Let us look at the worst-case scenario, a sequence of Is. In this

case, the signal element pattern +VO - VO is repeated every 4 bits. A nonperiodic signal has changed to a periodic signal with the period equal to 4 times the bit duration. This worst case situation can be simulated as an analog signal with a frequency one-fourth of the bit rate. In other words, the signal rate for MLT-3 is one-fourth the bit rate. This makes MLT-3 a suitable choice when we need to send 100 Mbps on a copper wire that cannot support more than 32 MHz (frequencies above this level create electromagnetic emissions).

Block Coding

We need redundancy to ensure synchronization and to provide some kind of inherent error detecting. Block coding can give us this redundancy and improve the performance of line coding. In general, block coding changes a block of m bits into a block of n bits, where n is larger than in. Block coding is referred to as an mB/nB encoding technique.

The slash in block encoding (for example, 4B/5B) distinguishes block encoding from multilevel encoding (for example, 8B6T), which is written without a slash. Block coding normally involves three steps: division, substitution, and combination. In the division step, a sequence of bits is divided into groups of in bits. For example, in 4B/5B encoding, the original bit sequence is divided into 4-bit groups. The heart of block coding is the substitution step. In this step, we substitute an m-bit group for an n-bit group. For example, in 4B/5B encoding we substitute a 4-bit code for a 5-bit group. Finally, the n-bit groups are combined together to form a stream. The new stream has more bits than the original bits. Figure 1.34 shows the procedure.

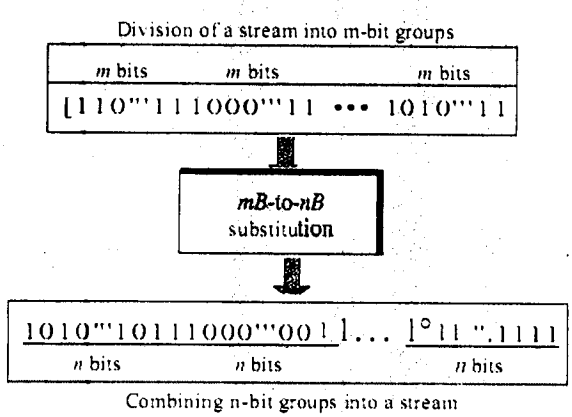


Figure 1.34 Block coding concept

4B/5B

The 4B/5B coding scheme was designed to be used in combination with NRZ-I. Recall that NRZ-I has a good signal rate, one-half that of the biphase, but it has a synchronization problem. A long sequence of as can make the receiver clock lose synchronization. One solution is to change the bit stream, prior to encoding with NRZ-I, so that it does not have a long stream of as 1. The 4B/5B scheme achieves this goal. The block-coded stream does not have more than three consecutive as, as we will see later. At the receiver, the NRZ-I encoded digital signal is first decoded into a stream of bits and then decoded to remove the redundancy. Figure 1.35 shows the idea.

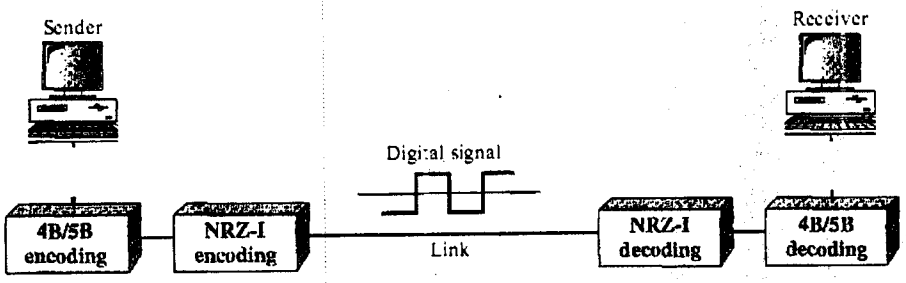


Figure 1.35 Using block coding 4B/5B with NRZ-I line coding scheme

In 4B/5B, the 5-bit output that replaces the 4-bit input has no more than one leading zero (left bit) and no more than two trailing zeros (right bits). So when different groups are combined to make a new sequence, there are never more than three consecutive as. (Note that NRZ-1 has no problem with sequences of Is.) Table 1.2 shows the corresponding pairs used in 4B/5B encoding. Note that the first two columns pair a 4-bit group with a 5-bit group. A group of 4 bits can have only 16 different combinations while a group of5 bits can have 32 different combinations. This means that there are 16 groups that are not used for 4B/5B encoding. Some of these unused groups are used for control purposes; the others are not used at all. The latter provide a kind of error detection. If a 5-bit group arrives that belongs to the unused portion of the table, the receiver knows that there is an error in the transmission.

Data Sequence	*Encoded Sequence*	*Control Sequence*	*Encoded Sequence*
0000	11110	Q (Quiet)	00000
0001	01001	I (Idle)	11111
0010	10100	H (Halt)	00100
0011	10101	J (Start delimiter)	11000
0100	01010	K (Start delimiter)	10001
0101	01011	T (End delimiter)	01101
0110	01110	S (Set)	11001
0111	01111	R (Reset)	00111
1000	10010		
1001	10011		
1010	10110		
1011	10111		
1100	11 010		
1101	11011		
1110	11100		
1111	11101		

Table 1.2 4B/5B mapping codes

Figure 1.36 shows an example of substitution in 4B/5B coding. 4B/5B encoding solves the problem of synchronization and overcomes one of the deficiencies of NRZ-I, however, we need to remember that it increases the signal rate of NRZ-I. The redundant bits add 20 percent more baud. Still, the result is less than the biphase scheme which has a signal rate of 2 times that of NRZ-I, however. 4B/5B block encoding does not solve the DC component problem of NRZ-I. If a DC component is unacceptable, we need to use biphase or bipolar encoding.

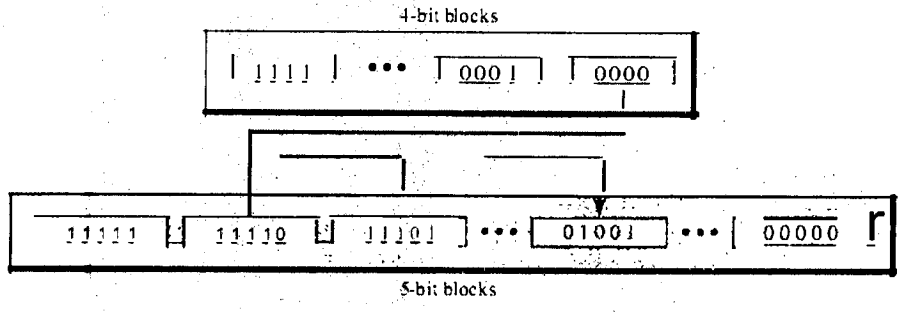


Figure 1.36 Substitution in 48/5B block coding

1.6 Error Detection and Correction

Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting errors.

Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy.

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. In a single-bit error, a 0 is changed to a 1 or a 1 to a 0. In a burst error, multiple bits are changed. For example, a 11100s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the 12 bits of information.

Single Bit Error

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.

Figure 1.37 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In Figure 1.37, 000000 10 (ASCII STX) was sent, meaning start of text, but 00001010 (ASCII LF. was received, meaning line feed).

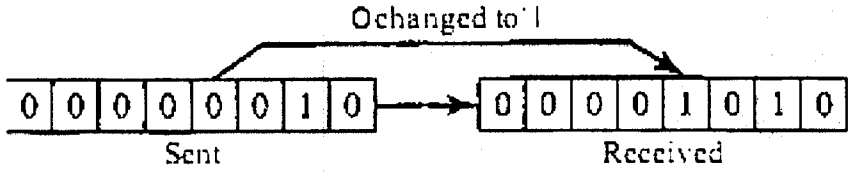


Figure 1.37 Single-bit error

Single-bit errors are the least likely type of error in serial data transmission. To understand imagine data sent at 1Mbps. This means that each bit lasts only 1/1,000,000 s, or 1µs. For a single-bit error to occur the noise must have duration of only 1µs, which is very rare; noise normally lasts much longer than this.

Burst Error

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 1.38 shows the effect of a burst error on a data unit. In this case, 0100010001000011 were sent, but 0101110101100011 were received. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

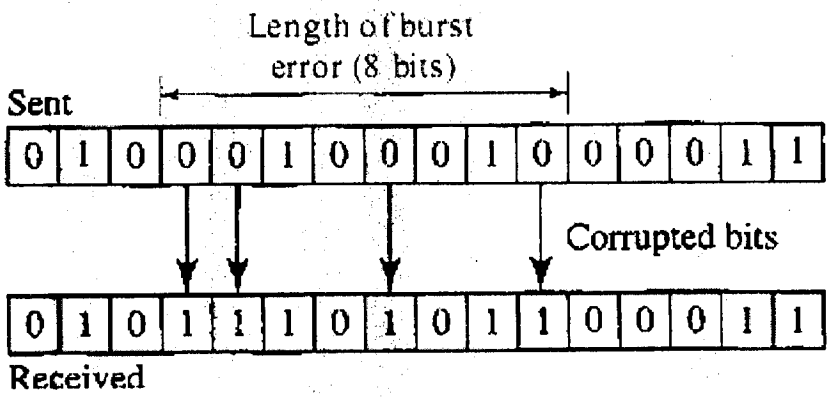


Figure 1.38 Burst error of length 8

A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1kbps, a noise of 11100 s can affect 10 bits; if we are sending data at 1Mbps, the same noise can affect 10000 bits.

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 looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. 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 the errors and the size of the message are important factors. If we need to correct one 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 possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Forward Error Correction versus Retransmission

There are two main methods of error correction. Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small. Correction by retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free (usually, not all errors can be detected).

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors. The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme. Figure 1.39 shows the general idea of coding.

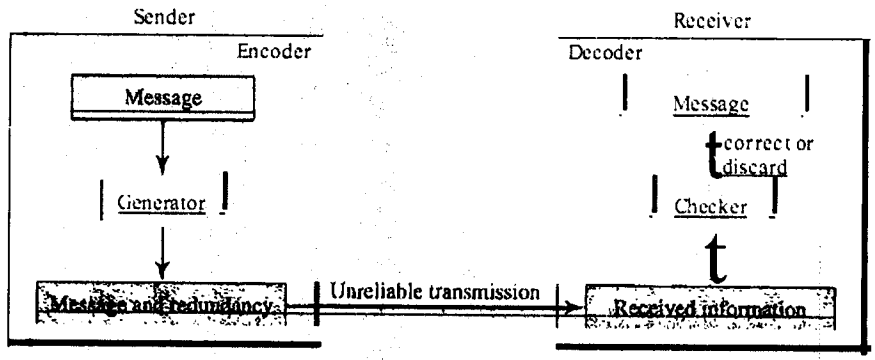


Figure 1.39 The structure of encoder and decoder

Modular Arithmetic

Before we finish this section, let us briefly discuss a concept basic to computer science in general and to error detection and correction in particular: modular arithmetic. Our intent here is not to delve deeply into the mathematics of this topic; we present just enough information to provide a background to materials discussed in this chapter.

In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus N. We then use only the integers 0 to N - 1, inclusive. This is modulo-N arithmetic. For example, if the modulus is 12, we use only the integers 0 to 11, inclusive. An example of modulo arithmetic is our clock system. It is based on modulo- 12 arithmetic, substituting the number 12 for 0. In a modulo-N system, if a number is greater than N, it is divided by N and the remainder is the result. If it is negative, as many Ns as needed are added to make it positive. Consider our clock system again. If we start a job at 11 A.M. and the job takes 5 h, we can say that the job is to be finished at 16:00 if we are in the military, or we can say that it will be finished at 4 P.M. (the remainder of 16/12 is 4).

Addition and subtraction in modulo arithmetic are simple. There is no carry when you add two digits in a column. There is no carry when you subtract one digit from another in a column.

Modulo -2 Arithmetic

Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus N is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.

Adding:	0+0=0	0+1=1	1+0=1	1+1=0
Subtracting:	0-0=0	0-1=1	1-0=1	1-1=0

Notice particularly that addition and subtraction give the same results. In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is I if two bits are different. Figure 1.40 shows this operation.

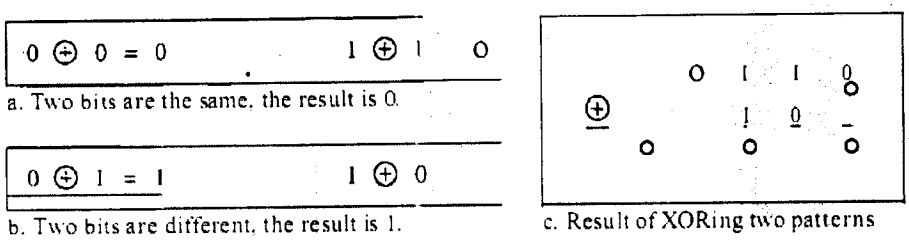


Figure 1.40 XORing of two single bits or two words

Block Coding

In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length ii = k + r. The resulting n-bit blocks are called code words, how the extra r bits is chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of data words, each of size k, and a set of code words, each of size of ii. With k bits, we can create a combination of 2^k data words; with n bits, we can create a combination of 2^n code words. Since n > k, the number of possible code words is larger than the number of possible data words. The block coding process is one-to-one; the same data word is always encoded as the same codeword. This means that we have $2^n - 2^k$ code words that are not used. We call these code words invalid or illegal. Figure 1.41 shows the situation.

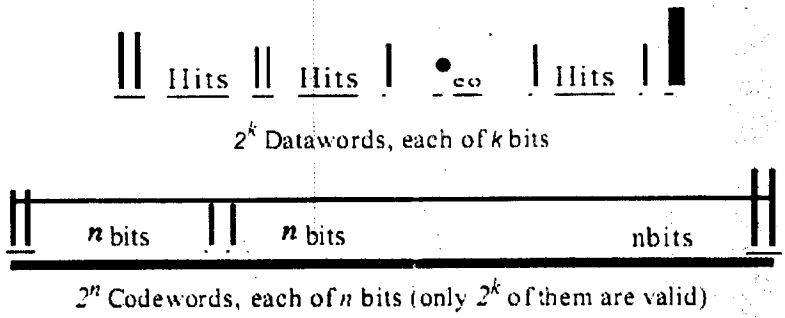


Figure 1.41 Data words and code words in block coding

Error Detection

How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid code words.

2. The original codeword has changed to an invalid one.

Figure 1.42 shows the role of lock coding in error detection.

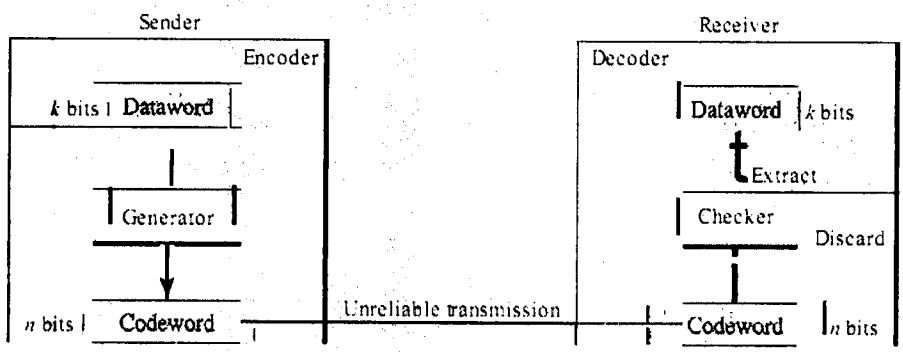


Figure 1.42 Process of error detection in block coding

The sender creates code words out of data words by using a generator that applies the rules and procedures of encoding (discussed later). Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid code words, the word is accepted; the corresponding data word is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected. This type of coding can detect only single errors. Two or more errors may remain undetected.

Error Correction

As we said before, error correction is much more difficult than error detection. In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent. We can say that we need more redundant bits for error correction than for error detection. Figure 1.43

shows the role of block coding in error correction. We can see that the idea is the same as error detection but the checker functions are much more complex.

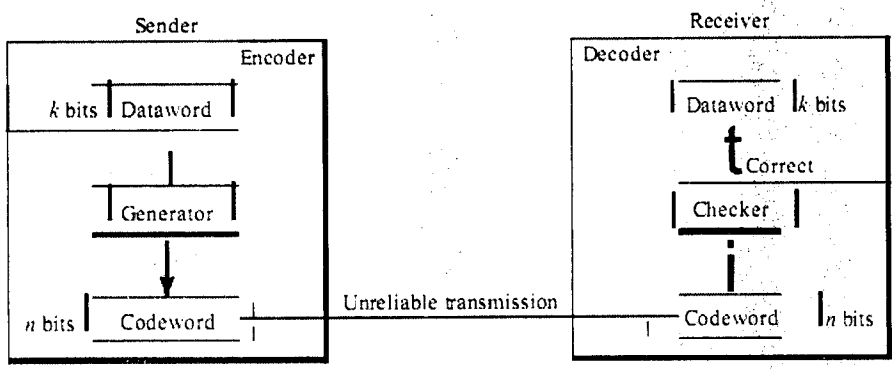


Figure 1.43 Structure of encoder and decoder in error correction

Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as d(x, y,).

The hamming distance can easily be found if we apply the XOR operation (⊕) on the two words and count the number of 1s in the result. Note that the hamming distance is a value greater than zero.

Minimum Hamming Distance

Although the concept of the hamming distance is the central point in dealing with error detection and correction codes, the measurement that is used for designing a code is the minimum hamming distance. In a set of words, the minimum hamming distance is the smallest hamming distance between all possible pairs. We use to define the minimum hamming distance in a coding scheme. To find this value, we find the hamming distances between all words and select the smallest one.

Hamming Distance and Error

Before we explore the criteria for error detection or correction, let us discuss the relationship between the Hamming distance and errors occurring during transmission. When a codeword is corrupted during

transmission, the Hamming distance between the sent and received code words is the number of bits affected by the error. In other words, the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is d (00000, 01101) 3.

Minimum Distance for Error Detection

Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to detect up to s errors, the minimum distance between the valid codes must be s + 1, so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid code words is s + 1, the received codeword cannot be erroneously mistaken for another codeword. The distances are not enough (s + 1) for the receiver to accept it as valid. The error il1 be detected. We need to clarify a point here: Although a code with d_{min} = s + 1 may be able to detect more than s errors in some special cases, only s or fewer errors are guaranteed to be detected.

We can look at this geometrically. Let us assume that the sent codeword x is at the center of a circle with radius s. All other received code words that are created by 1 to s errors are points inside the circle or on the perimeter of the circle. All other valid code words must be outside the circle, as shown in Figure 1.44.

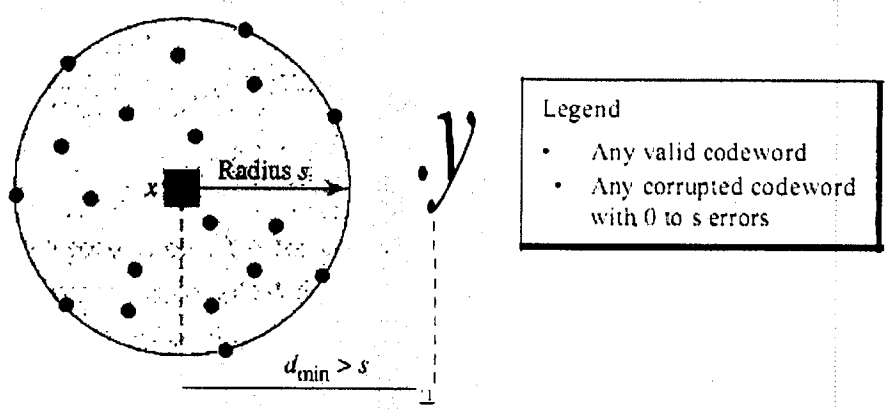


Figure 1.44 Geometric concept for finding d_{min} in error detection

Minimum Distance for Error Correction

Error correction is more complex than error detection; a decision is involved. When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword. Each valid codeword has its own territory.

We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius oft and that the valid codeword is at the center. For example, suppose a codeword x is corrupted by t bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the center. Note that we assume that only up to i errors have occurred; otherwise, the decision is wrong. Figure 1.45 shows this geometric interpretation. Some texts use a sphere to show the distance between all valid block codes.

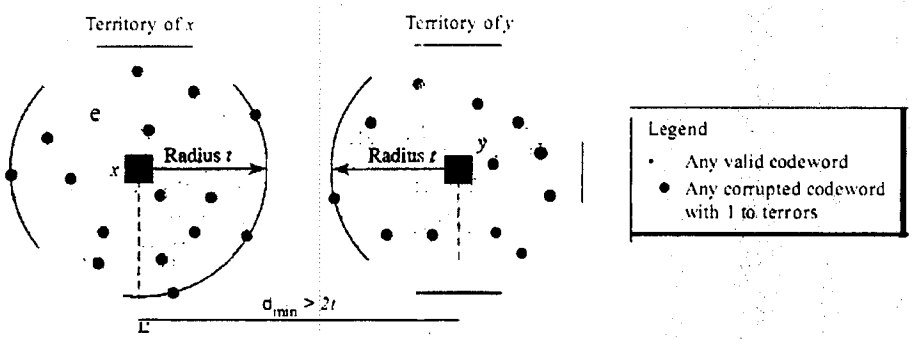


Figure 1.45 Geometric concept for finding d_{min} in error correction

In Figure 1.45, $d_{min} > 2t$; since the next integer increment is 1, we can say that $d_{min} = 2t + 1$.

Linear Block Codes

Almost all block codes used today belong to a subset called linear block codes. The use of nonlinear block codes for error detection and correction is not as widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes.

The formal definition of linear block codes requires the knowledge of abstract algebra (particularly Galois fields), which is beyond the scope of this book. We therefore give an informal definition. For our purposes, a linear block code is a code in which the exclusive OR (addition modulo-2) of two valid code words creates another valid codeword.

Minimum Distance for Linear Block Codes

It is simple to find the minimum hamming distance for a linear block code. The minimum hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Some Linear Block Codes

Let us now show some linear block codes. These codes are trivial because we can easily find the encoding and decoding algorithms and check their performances.

Simple Parity- Check Code

Perhaps the most familiar error-detecting code is the simple parity-check code. In this code, a k-bit data word is changed to an n-bit codeword where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of Is, we discuss the even case. The minimum Hamming distance for this category is d_{min} = 2, which means that the code is a single-bit error-detecting code; it cannot correct any error.

Figure 1.46 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver).

The encoder uses a generator that takes a copy of a 4-bit data word (a_0 a_1 a_2 and a_3, and generates a parity bit r0. The data word bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even.

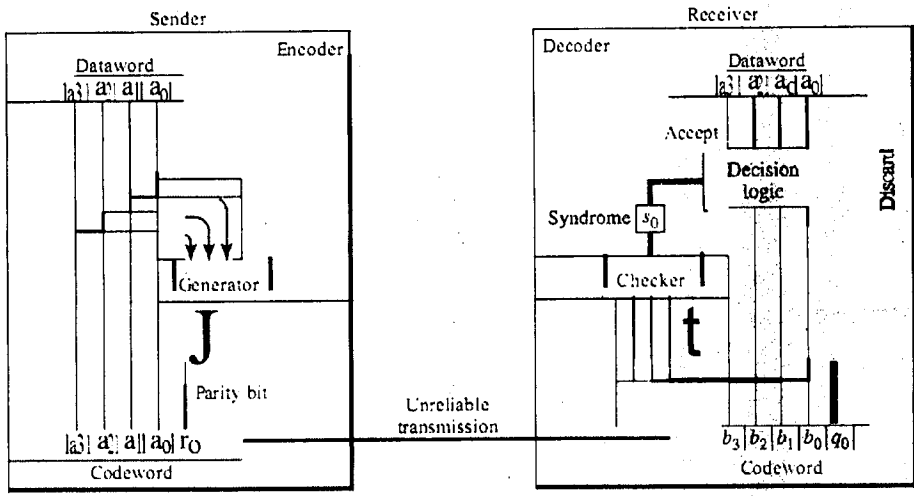


Figure 1.46 Encoder and decoder for simple parity-check code

This is normally done by adding the 4 bits of the data word (modulo-2); the result is the parity bit. in other words,

$r_0 = a_3 + a_2 + a_1 + a_0$ (modulo-2)

If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of is in the codeword is even.

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

$S_0 = b_3 + b_2 + b_1 + b_0 + q_0$ (modulo-2)

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

A better approach is the two-dimensional parity check. In this method, the data- word is organized in a table (rows and columns). In Figure 1.47, the data to be sent, five 7-bit bytes, are put in separate rows. For each row and each column, 1 parity-check bit is calculated. The whole table is then sent

to the receiver, which finds the syndrome for each row and each column. As Figure 1.47 shows, the two-dimensional parity check can detect up to three errors that occur anywhere in the table (arrows point to the locations of the created nonzero syndromes). However, errors affecting 4 bits may not be detected.

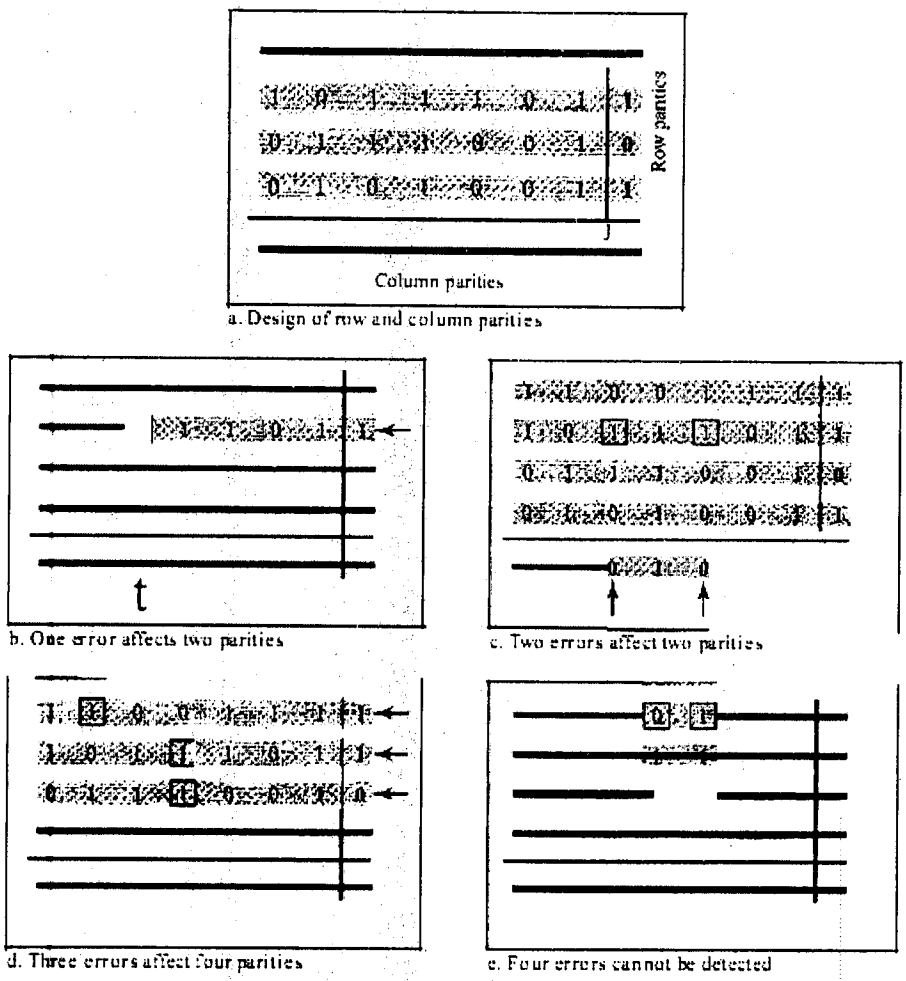


Figure 1.47 Two-dimensional parity-check code

Hammining Codes

Now let us discuss a category of error-correcting codes called Hamming codes. These codes were originally designed with $d_{min} = 3$, which means that they can detect up to two errors or correct one single error. Although there are some Hamming codes that can correct more than one error, our discussion focuses on the single-bit error-correcting code. First let us find

the relationship between n and k in a hamming code. We need to choose an integer m >= 3. The values of n and k are then calculated from mas $n = 2^m - 1$ and k = n - m. The number of check bits r = m.

Figure 1.48 shows the structure of the encoder and decoder for this example.

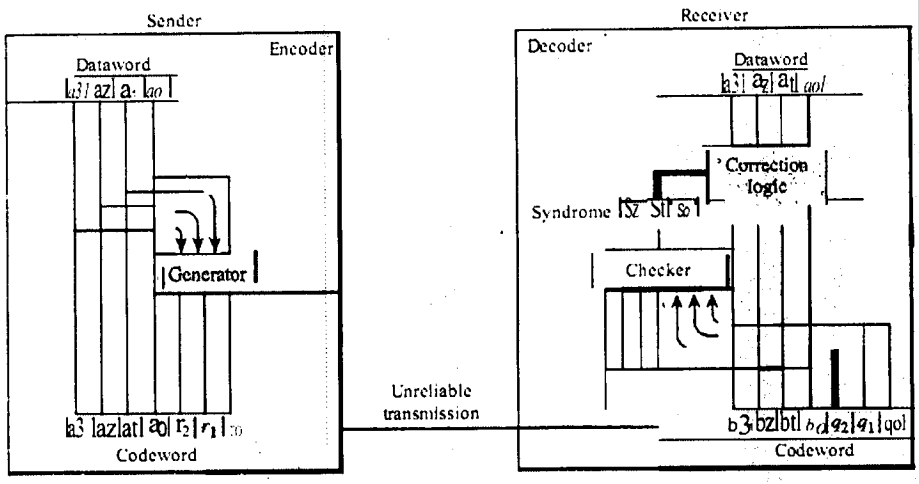


Figure 1.48 The structure of the encoder and decoder for a Hamming code

A copy of a 4-bit data word is fed into the generator that creates three parity checks r_0, r_1 and r_2 as shown below:

$$r_0 = a_2 + a_1 + a_0 \text{ modulo-2}$$

$$r_1 = a_3 + a_2 + a_1 \text{ modulo-2}$$

$$r_2 = a_1 + a_0 + a_3 \text{ modulo-2}$$

In other words, each of the parity-check bits handles 3 out of the 4 bits of the data word. The total number of is in each 4-bit combination (3 data word bits and 1 parity bit) must be even. We are not saying that these three equations are unique; any three equations that involve 3 of the 4 bits in the data word and create independent equations (a combination of two cannot create the third) are valid.

The checker in the decoder creates a 3-bit syndrome (s2s1s0) in which each bit is the parity check for 4 out of the 7 bits in the received codeword:

So = bz+b i+bo+qo modulo-2

S1 = b3+bz+bj +ql modulo-2

S2 = b1 +b0+b3+qz modulo-2

The equations used by the checker are the same as those used by the generator with the parity-check bits added to the right-hand side of the equation. The 3-bit syndrome creates eight different bit patterns (000 to 111) that can represent eight different conditions. These conditions define a lack of error or an error in 1 of the 7 bits of the received codeword. as shown in Table 1.3.

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	ql	b_2	q_2	b_0	b_3	b_1

Table 1.3 Logical decision made by the correction logic analyzer a/the decoder

Note that the generator is not concerned with the four cases shaded in Table 1.3 because there is either no error or an error in the parity bit. In the other four cases, 1 of the bits must be flipped (changed from 0 to 1 or 1 to 0) to find the correct data word.

The syndrome values in Table 1.3 are based on the syndrome bit calculations. For example, if q0 is in error, S0 is the only bit affected; the syndrome, therefore, is 001. If b2 is in error, S0 and S1 are the bits affected; the syndrome therefore is 01l. Similarly, if b1 is in error, all 3. syndrome bits are affected and the syndrome is 111.

There are two points we need to emphasize here. First, if two errors occur during transmission, the created data word might not be the right one. Second, if we want to use the above code for error detection, we need a different design.

Performance

A Hamming code can only correct a single error or detect a double error. However, there is a way to make it detect a burst error, as shown in Figure 1.49.

The key is to split a burst error between several code words, one error for each codeword. In data communications, we normally send a packet or a

frame of data. To make the Hamming code respond to a burst error of size N, we need to make N code words out of our frame. Then, instead of sending one codeword at a time, we arrange the code words in a table and send the bits in the table a column at a time. In Figure 1.49, the bits are sent column by column (from the left). I n each column, the bits are sent from the bottom to the top. I n this way, a frame is made out of the four code words and sent to the receiver. Figure 1.49 shows that when a burst error of size 4 corrupts the frame, only 1 bit from each codeword is corrupted. The corrupted bit in each codeword can then easily be corrected at the receiver.

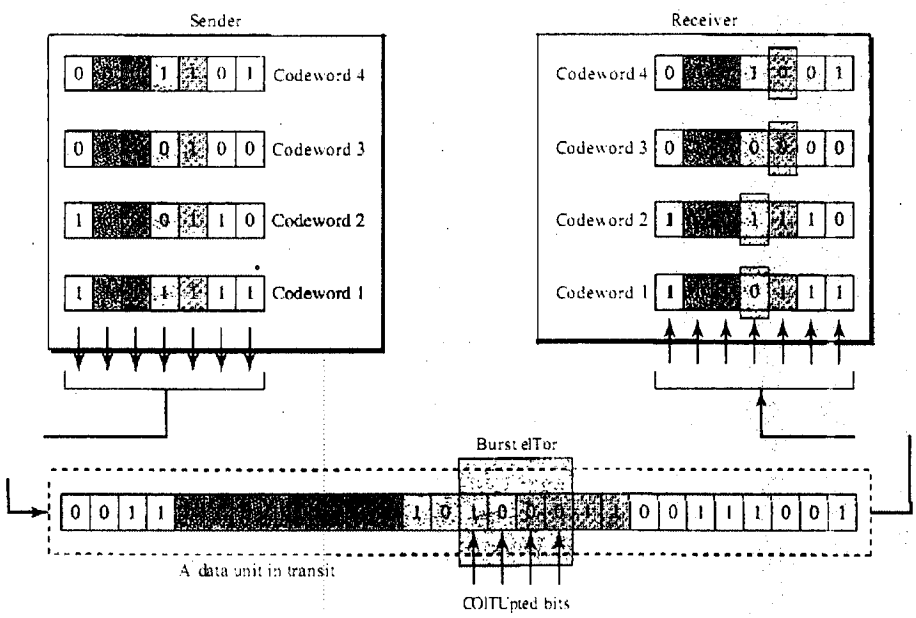


Figure 1.49 Burst error correction using Hamming code

Cyclic Codes

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 are a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word a_0 to a_6 and the bits in the second word b0to b6, we can shift the bits by using the following:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a category of cyclic codes called the cyclic redundancy check (CRC) that is used in networks such as LANs and WANs.

Table 1.4 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Table 1.4 A CRC code with C(7, 4)

Figure 1.50 shows one possible design for the encoder and decoder.

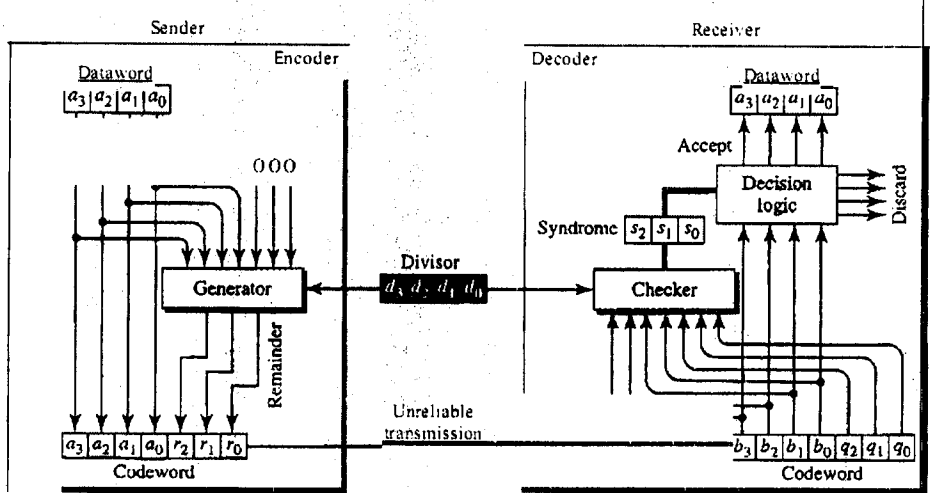


Figure 1.50 CRC encoder and decoder

In the encoder, the data word has k bits (4 here); the codeword has ii bits (7 here). The size of the data word is augmented by adding n - k (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n - k + 1 (4 here), predefined and agreed upon. The generator divides the augmented data word by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r2r1r0,) is appended to the data word to create the codeword.

The decoder receives the possibly corrupted codeword. A copy of all ii bits is fed to the checker which is a replica of the generator. The remainder produced by the checker is a syndrome of n - k (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits arc all as, the 4 leftmost bits of the code word arc accepted as the data word (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes the data word and augments it with n - k number of 0s. It then divides the augmented data word by the divisor, as shown in Figure 1. 51.

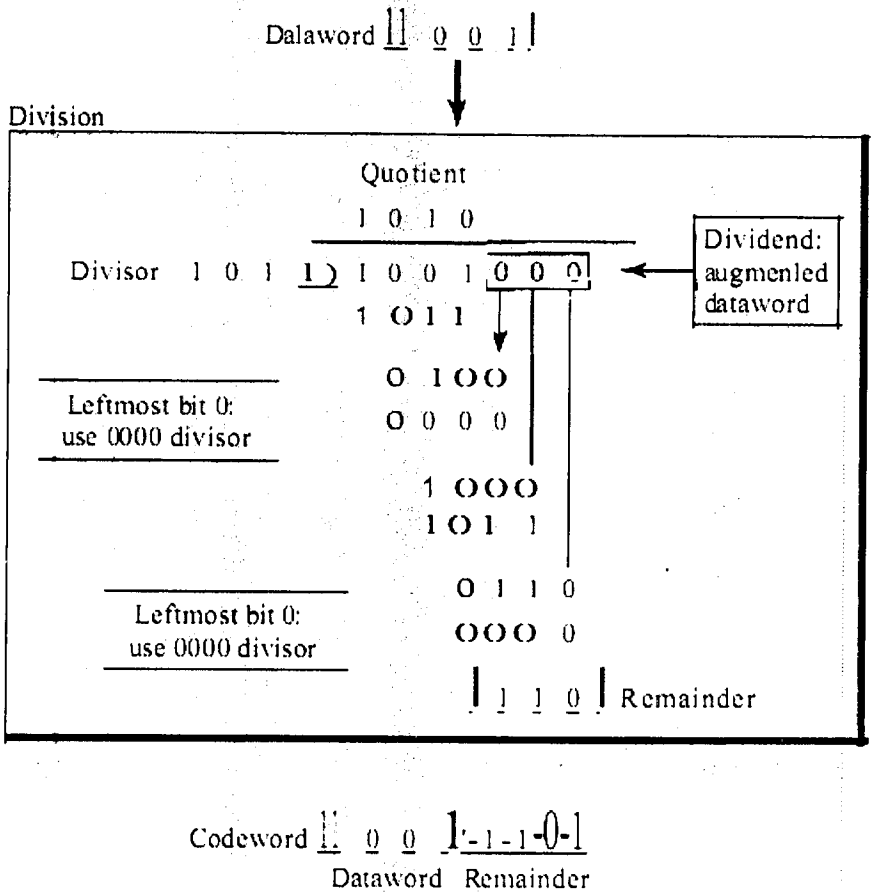


Figure 1.51 Division in CRC encode

The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. However, in this case addition and subtraction are the same. We use the XOR operation to do both.

As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. I f the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor.

When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits (r2 r1 and r0. They are appended to the data word to create the codeword.

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; the data word is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 1.52 shows two cases: The left- hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all 0s (it is 011).

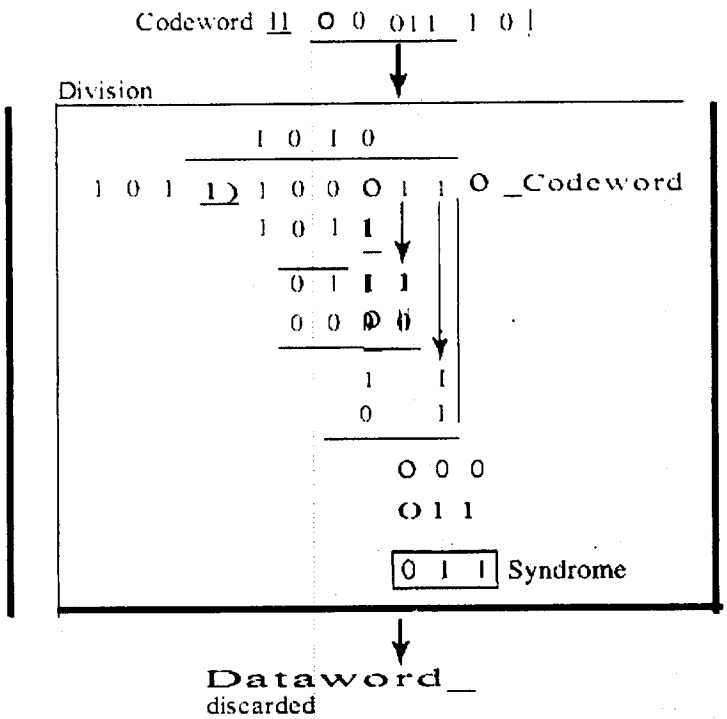


Figure 1.52 Division in the CRC decoder for two cases

Hardware Implementation

One of the advantages of a cyclic code is that the encoder and decoder can easily and cheaply be implemented in hardware by using a handful of electronic devices. Also, a hardware implementation increases the rate of check bit and syndrome bit calculation. In this section, we try to show, step by step, the process.

Divisor

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

1. The divisor is repeatedly XORed with part of the dividend.
2. The divisor has n - k + 1 bits which either are predefined or are all. In other words, the bits do not change from one data word to another. In our previous example, the divisor bits were either 1011 or 0000. The choice was based on the leftmost bit of the part of the augmented data bits that are active in the XOR operation.
3. A close look shows that only n – k bits of the divisor are needed in the XOR operation. The leftmost bit is not needed because the result of the operation is always 0, no matter what the value of this bit. The reason is that the inputs to this XOR operation are either both 0s and both 1s. In our previous example, only 3 bits, not 4, is actually used in the XOR operation.

Using these points, we can make a fixed (hardwired) divisor that can be used for a cyclic code if we know the divisor pattern. Figure 1.53 shows such a design for our previous example. We have also shown the XOR devices used for the operation.

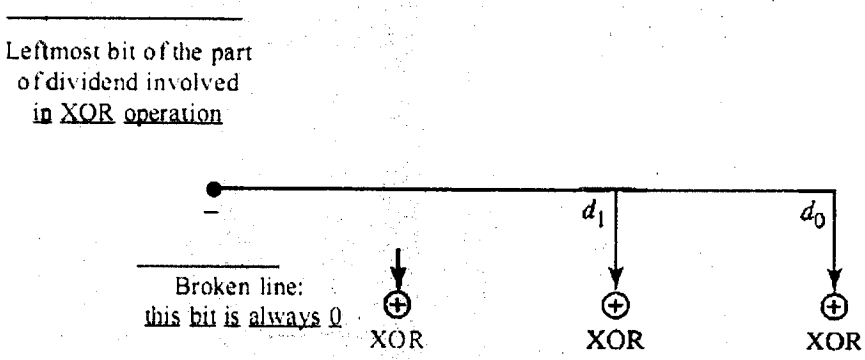


Figure 1.53 Hardwired design of the divisor in CRC

Note that if the leftmost bit of the part of dividend to be used in this step is 1, the divisor bits (d2d1d0), are all; if the leftmost bit is 0, the divisor bits arc 000. The design provides the right choice based on the leftmost bit.

Augmented Data word

In our paper-and-pencil division process in Figure 1.51, we show the augmented data- word as fixed in position with the divisor bits shifting to the right, 1 bit in each step. The divisor bits are aligned with the appropriate part of the augmented data word. Now that our divisor is fixed, we need instead to shift the bits of the augmented data word to the left (opposite direction) to align the divisor bits with the appropriate part. There is no need to store the augmented data word bits.

Remainder

In our previous example, the remainder is 3 bits (n - k bits in general) in length. We can use three registers (single-bit storage devices) to hold these bits. To find the final remainder of the division, we need to modify our division process. The following is the step-by-step process that can be used to simulate the division process in hardware (or even in software).

1. We assume that the remainder is originally all 0s (000 in our example).

2. At each time click (arrival of 1 bit from an augmented data word), we repeat the following two actions:

    a. We use the leftmost bit to make a decision about the divisor (011 or 000).

    b. The other 2 bits of the remainder and the next bit from the augmented data word (total of 3 bits) are XORed with the 3-bit divisor to create the next remainder.

Figure 1.54 shows this simulator, but note that this is not the final design; there will be more improvements.

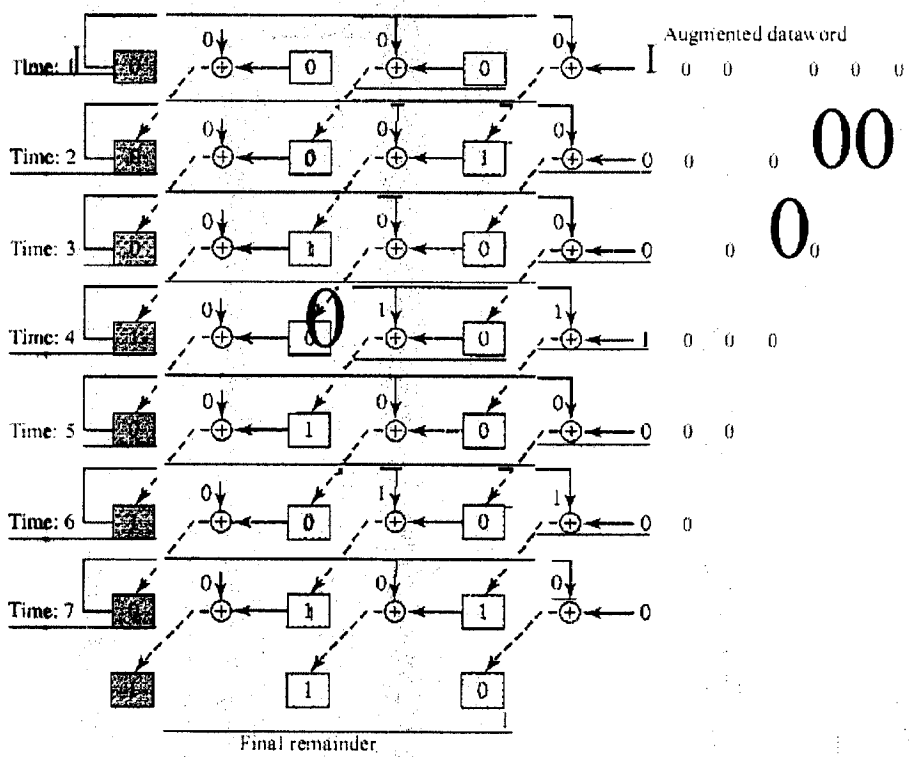


Figure 1.54 Simulation of division in CRC encoder

At each clock tick, shown as different times, one of the bits from the augmented data word is used in the XOR process. If we look carefully at the design, we have seven steps here, while in the paper-and-pencil method we had only four steps. The first three steps have been added here to make each step equal and to make the design for each step the same. Steps 1, 2, and 3 push the first 3 bits to the remainder registers; steps 4, 5, 6, and 7 matches the paper-and-pencil design. Note that the values in the remainder register in steps 4 to 7 exactly match the values in the paper-and-pencil design. The final remainder is also the same.

The above design is for demonstration purposes only. It needs simplification to be practical. First, we do not need to keep the intermediate values of the remainder bits; we need only the final bits. We therefore need only 3 registers instead of 24. After the XOR operations, we do not need the bit values of the previous remainder. Also, we do not need 21 XOR devices; two are enough because the output of an XOR operation in which one of the bits is 0 is simply the value of the other bit. This other bit can be

used as the output with these two modifications; the design becomes tremendously simpler and less expensive, as shown in Figure 1.55.

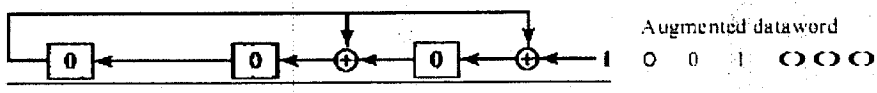


Figure 1.55 The CRC encoder design using shift registers

We need, however, to make the registers shift registers. A 1-bit shift register holds a bit for duration of one clock time. At a time click, the shift register accepts the bit at its input port, stores the new bit, and displays it on the output port. The content and the output remain the same until the next input arrives. When we connect several l-bit shift registers together, it looks as if the contents of the register are shifting.

General Design

A general design for the encoder and decoder is shown in Figure 1.56.

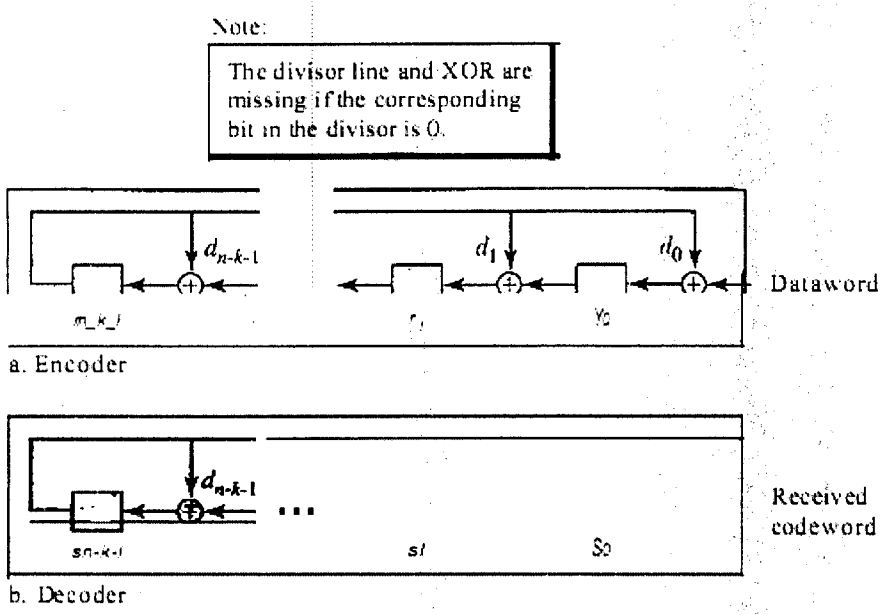


Figure 1.56 General design of encoder and decoder of a CRC code

Note that we have n - k 1-bit shift registers in both the encoder and decoder. We have up to n - k XOR devices, but the divisors normally have several 0s in their pattern, which reduces the number of devices. Also note that, instead of augmented data words, we show the data word itself as the

input because after the bits in the data word are all fed into the encoder, the extra bits, which all are 0s, do not have any effect on the rightmost XOR. Of course, the process needs to be continued for another n - k steps before the check bits are ready. This fact is one of the criticisms of this design. Better schemes have been designed to eliminate this waiting time (the check bits are ready after k steps), but we leave this as a research topic for the reader. In the decoder, however, the entire codeword must be fed to the decoder before the syndrome is ready.

Polynomials

A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. Again, this section is optional.

A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Figure 1.57 shows a binary pattern and its polynomial representation. In Figure 1.57a we show how to translate a binary pattern to a polynomial; in Figure 1.57b we show how the polynomial can be shortened by removing all terms with zero coefficients and replacing xl by x and x0 by 1.

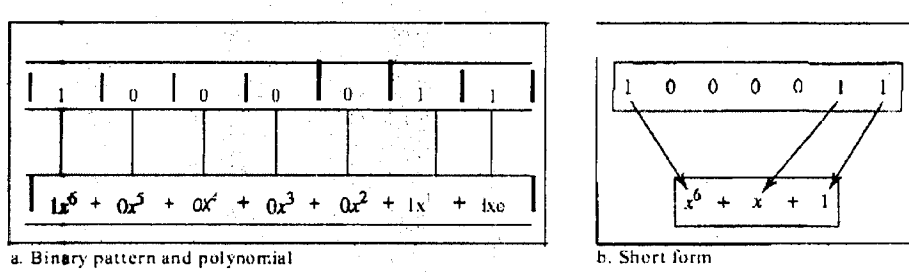


Figure 1.57 A polynomial to represent a binary word

Figure 1.57 shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x^{23} + x^3 + 1$. Here the bit pattern is 24 bits in length (three Is and twenty one 0s) while the polynomial is just three terms.

Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x^6 + x + 1$ is 6. Note that the degree of a polynomial is 1 less that the number of bits in the pattern. The bit pattern in this case has 7 bits.

Adding and Subtracting Polynomials

Adding and subtracting polynomials in mathematics are done by adding or subtracting the coefficients of terms with the same power. In our case, the coefficients are only 0 and 1, and adding is in modulo-2. This has two consequences. First, addition and subtraction are the same. Second, adding or subtracting is done by combining terms and deleting pairs of identical terms. For example, adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ gives just $x^6 + x^5$. The terms x^4 and x^2 are deleted. However, note that if we add, for example, three polynomials and we get x^2 three times, we delete a pair of then and keep the third.

Multiplying or Dividing Terms

In this arithmetic, multiplying a term by another term is very simple; we just add the powers. For example, $x^3 \times x^4$ is x^7, for dividing, we just subtract the power of the second term from the power of the first. For example, x^5/x^2 is x^3.

Multiplying Two Polynomials

Multiplying a polynomial by another is done term by term. Each term of the first polynomial must be multiplied by all terms of the second. The result, of course, is then simplified, and pairs of equal term are deleted. The following is an example:

$$(x^5 + x^3 + x^2 + x)(x^2 + x + 1)$$

$$= x^7 + x^6 + x^5 + x^5 + x^4 + x^3 + x^2 + x^3 + x^2 + x$$

$$= x^7 + x^6 + x^3 + x$$

Dividing One Polynomial by Another

Division of polynomials is conceptually the same as the binary division we discussed for an encoder. We divide the first term of the dividend by the first term of the divisor to get the first term of the quotient. We multiply the term in the quotient by the divisor and subtract the result from the dividend. We repeat the process until the dividend degree is less than the divisor degree. We will show an example of division later in this chapter.

Shifting

A binary pattern is often shifted a number of bits to the right or left. Shifting to the left means adding extra 0s as rightmost bits; shifting to the right means deleting some rightmost bits. Shifting to the left is accomplished by multiplying each term of the polynomial by m, where in is the number of shifted bits; shifting to the right is accomplished by dividing each term of the polynomial by x^m. The following shows shifting to the left and to the right. Note that we do not have negative powers in the polynomial representation.

Shifting left 3 bits: 10011 becomes 10011000 $x^4+x + 1$ becomes x^7

Shifting right 3 bits: 10011 becomes 10 $x^4+x + 1$ becomes x

When we augmented the data word in the encoder of Figure 1.51, we actually shifted the bits to the left. Also note that when we concatenate two bit patterns, we shift the first polynomial to the left and then add the second polynomial.

Cyclic Code Encoder Using Polynomials

Now that we have discussed operations on polynomials, we show the creation of a codeword from a data word. Figure 1.58 is the polynomial version of Figure 1.51. We can see that the process is shorter. The data word 1001 is represented as + 1. The divisor 1011 is represented as $x^3 + x + 1$. To find the augmented data word, we have left-shifted the data word 3 bits (multiplying by x^3). The result is $x^6 +x^3$. Division is straightforward. We divide the first term of the dividend, x^6, by the first term of the divisor, x^3. The first term of the quotient is then x^6/x^3, or x^3. Then we multiply x^3 by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend. The result is x^4, with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.

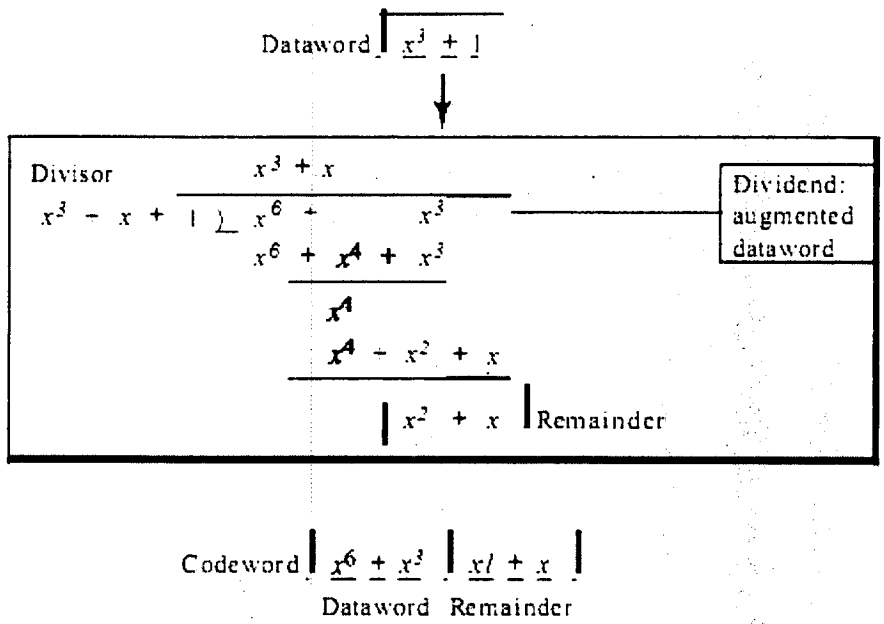


Figure 1.58 CRC division using polynomials

It can be seen that the polynomial representation can easily simplify the operation of division in this case, because the two steps involving all-0s divisors are not needed 1s. (Of course, one could argue that the all-0s divisor step can also be eliminated in binary division.) In a polynomial representation, the divisor is normally referred to as the generator polynomial t(x).

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

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking.

Like linear and cyclic codes, the checksum is based on the concept of redundancy. Several protocols still use the checksum for error detection as we will see in future chapters, although the tendency is to replace it with a CRC. This means that the CRC is also used in layers other than the data link layer.

Idea

The concept of the checksum is not difficult. Let us illustrate it with a few examples.

Example

Suppose our data 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 data are not accepted.

Example

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 1l, 12, 0, 6, -36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

The previous example has one major drawback. All of our data can be written as a 4-bit word (they are less than 15) except for the checksum. One solution is to use one's complement arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and 2^n - 1 using only n bits. If the number has more than n bits, the extra leftmost bits need to be added to the ii rightmost bits (wrapping). In one's complement arithmetic, a negative number can be represented by inverting all bits (changing a 0 to a 1 and a 1 to a 0). This is the same as subtracting the number from 2^n - 1.

1.7 Flow Control

As we saw in the previous section, frames are sometimes corrupted while in transit, with an error code like CRC used to detect such errors. While some error codes are strong enough also to correct errors, in practice the overhead is typically too large to handle the range of bit and burst errors that can he introduced on a network link. Even when error-correcting codes are used (e.g., on wireless links), some errors will he too severe to he corrected. As a result, some corrupt frames must be discarded. A link-level protocol that wants to deliver frames reliably must somehow recover from these discarded (lost) frames.

This is usually accomplished using a combination of two fundamental mechanisms acknowledgments and timeouts. An acknowledgment (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received an earlier frame. By control frame we mean a header without any data, although a protocol can piggyback an ACK on a data frame it just happens to be sending in the opposite direction. The receipt of an acknowledgment indicates to the sender of the original frame that its frame was successfully delivered. If the sender does not receive an acknowledgment after a reasonable amount of time, then it retransmits the original frame. This action of waiting a reasonable amount of time is called a timeout.

The general strategy of using acknowledgments and timeouts to implement reliable delivery is sometimes called automatic repeat request (normally abbreviated ARQ). This section describes three different ARQ algorithms using generic language; that is, we do not give detailed information about a particular protocol's header fields.

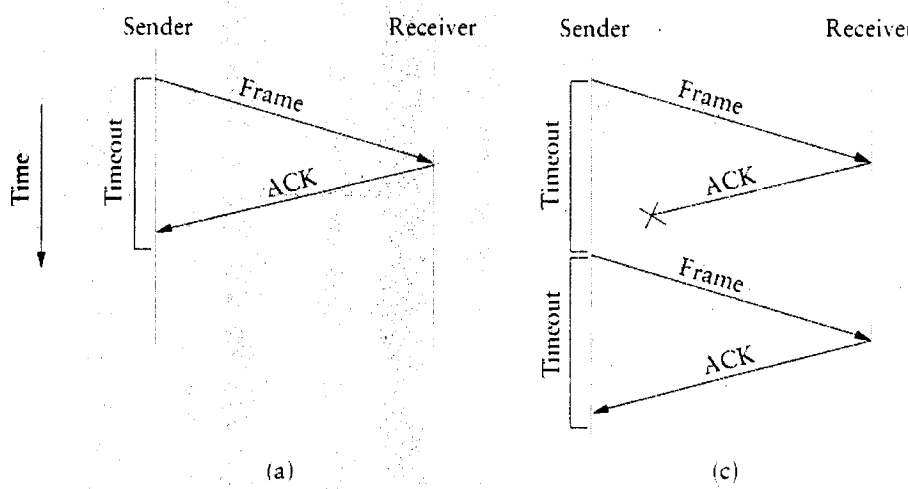
Stop-and-Wait

The simplest ARQ scheme is the stop-and-wait algorithm. The idea of stop-and-wait is straightforward: After transmitting one frame, the sender waits for an acknowledgment before transmitting the next frame. If the acknowledgment does not arrive after a certain period of time, the sender time out and retransmits the original frame.

Figure 1.59 illustrates four different scenarios that result from this basic algorithm. This figure is a timeline, a common way to depict a protocol's behavior. The sending side is represented on the left, the receiving side is

depicted on the right, and time flows from top to bottom. Figure 1.60(a) shows the situation in which the ACK is received before the timer expires,' (h) and (c) show the situation in which the original frame and the ACK, respectively, are lost, and (d) shows the situation in which the timeout fires too soon. Recall that by "lost" we mean that the frame was corrupted while in transit, that this corruption was detected by an error code on the receiver, and that the frame was subsequently discarded.

There is one important subtlety in the stop-and-wait algorithm. Suppose the sender sends a frame and the receiver acknowledges it, hut the acknowledgment is either lost or delayed in arriving. This situation is illustrated in timelines (c) and (d) of Figure 1.59. In both cases, the sender time out and retransmits the original frame, hut the receiver will think that it is the next frame, since it correctly received and acknowledged the first frame. This has the potential to cause duplicate copies of a frame to he delivered.



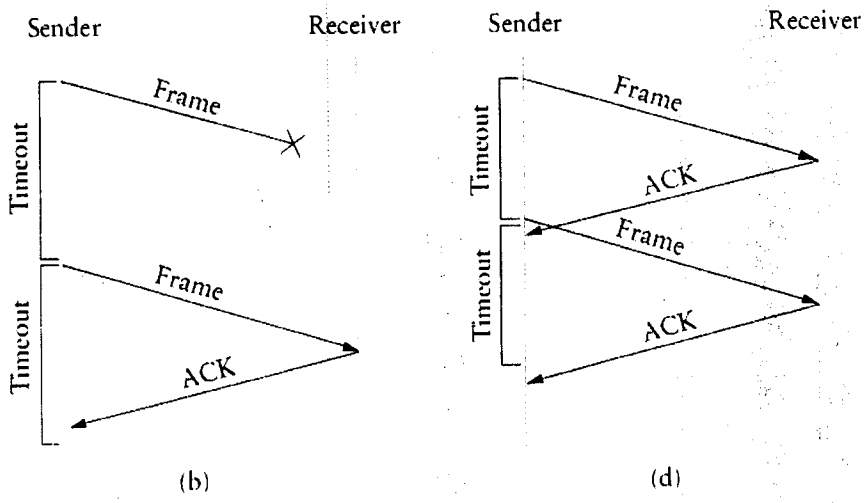


Figure 1.59 Timeline showing four different scenarios for the stop-and-wait algorithm. (a) The ACK is received before the timer expires; (b) the original frame is lost; (C) the ACK is lost; (d) the timeout fires too soon.

To address this problem, the header for a stop-and-wait protocol usually includes a 1-bit sequence number that is, the sequence number can take on the values 0 and 1and the sequence numbers used for each frame alternate, as illustrated in Figure 1.60. Thus, when the sender retransmits frame 0, the receiver can determine that it is seeing a second copy of frame 0 rather than the first copy of frame 1 and therefore can ignore it (the receiver still acknowledges it, in case the first ACK was lost).

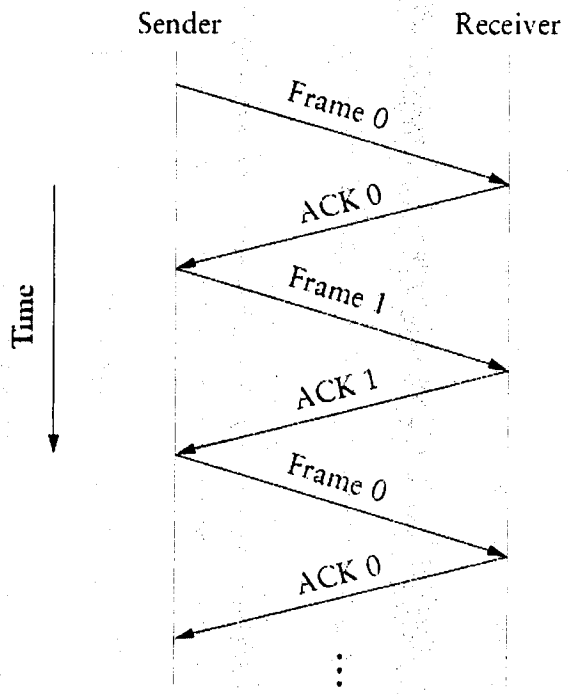


Figure 1.60 Timelineforstop-and-waitwith1-bitsequencenumber.

The main shortcoming of the stop-and-wait algorithm is that it allows the sender to have only one outstanding frame on the link at a time, and this may he far below the link's capacity. Consider, for example, a 1.5-Mbps link with a 45-ms round-trip time. This link has a delay x bandwidth product of 67.5 Kb, or approximately 8 KB. Since the sender can send only one frame per RTT, and assuming a frame size of 1 KB, this implies a maximum sending rate of

Bits per Frame ÷ Time Per Frame = 1024 x 8 ÷ 0.045 = 182 Kbps

or about one-eighth of the link's capacity. To use the link fully, then, we'd like the sender to be able to transmit up to eight frames before having to wait for an acknowledgment.

The significance of the bandwidth x delay product is that it represents the amount of data that could he in transit. We would like to be able to send this much data without waiting for the first acknowledgment. The principle at work here is often referred to as keeping the pipe full. The algorithms presented in the following two subsections do exactly this.

Sliding Window

Consider again the scenario in which the link has a delay x bandwidth product of 8 KB and frames are of 1-KB size. We would like the sender to be ready to transmit the ninth frame at pretty much the same moment that the ACK for the first frame arrives. The algorithm that allows us to do this is called sliding window, and an illustrative timeline is given in Figure 1.61.

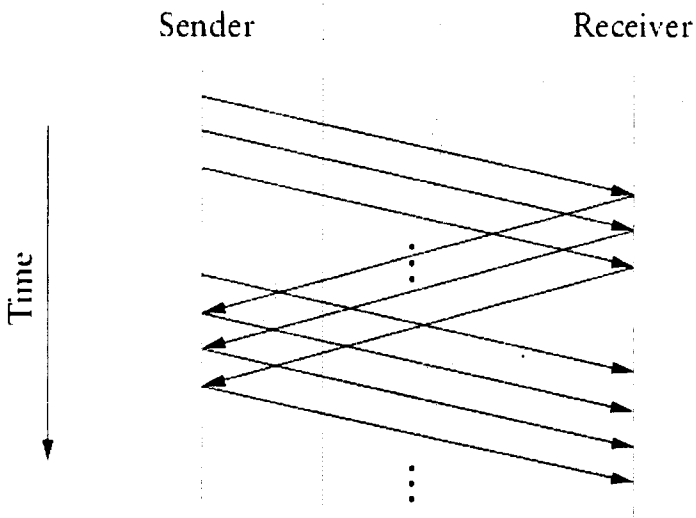


Figure 1.61 Time line for the sliding window algorithm.

The Sliding Window Algorithm

The sliding window algorithm works as follows. First, the sender assigns a sequence number, denoted SeqNum, to each frame. For now, let's ignore the fact that SeqNum is implemented by a finite-size header field and instead assume that it can grow infinitely large. The sender maintains three variables: The send window size, denoted SWS, gives the upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit; LAR denotes the sequence number of the last acknowledgment received; and LFS denotes the sequence number of the last frame sent. The sender also maintains the following invariant:

LFS — LAR ≤ SWS

This situation is illustrated in Figure 1.62

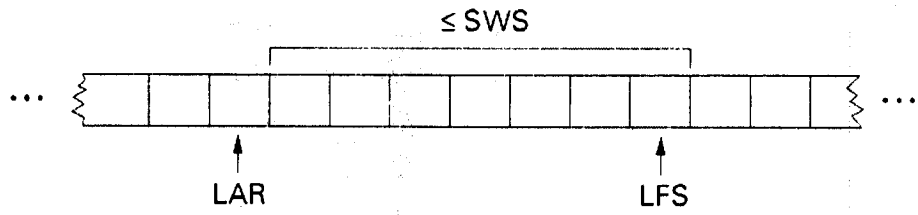


Figure 1.62 Sliding window on sender

When an acknowledgment arrives, the sender moves LAR to the right, thereby allowing the sender to transmit another frame. Also, the sender associates a timer with each frame it transmits, and it retransmits the frame should the timer expire before an ACK is received. Notice that the sender has to be willing to buffer up to SWS frames since it must be prepared to retransmit them until they are acknowledged.

The receiver maintains the following three variables: The receive window size, denoted RWS, gives the upper bound on the number of out-of-order frames that the receiver is willing to accept; LAF denotes the sequence number of the largest acceptable frame; and LFR denotes the sequence number of the last frame received. The receiver also maintains the following invariant:

LAF — LFR ≤ RWS

This situation is illustrated in Figure 1.63.

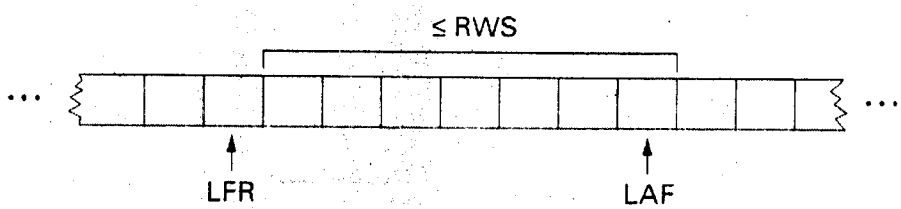


Figure 1.63 Sliding window on receiver.

When a frame with sequence number SeqNum arrives, the receiver takes the following action. If SeqNum ≤ LFR or SeqNum > LAF, then the frame is outside the receiver's window and it is discarded. if LFR < SeqNum ≤ LAF, then the frame is within the receiver's window and it is accepted. Now the receiver needs to decide whether or not to send an ACK. Let SeqNumToAck denote the largest sequence number not yet

acknowledged, such that all frames with sequence numbers less than or equal to SeqNumToAck have been received. The receiver acknowledges the receipt of SeqNumToAck, even if higher-numbered packets have been received.

This acknowledgment is said to be cumulative, it then sets LFR = SeqNumToAck and adjusts LAF LFR + RWS.

For example, suppose LFR = 5 (i.e., the last ACK the receiver sent was for sequence number 5), and RWS = 4. This implies that LAF = 9. Should frames 7 and 8 arrive, they will he buffered because they are within the receiver's window. However, no ACK needs to he sent since frame 6 is yet to arrive. Frames 7 and 8 are said to have arrived out of order. (Technically, the receiver could resend an ACK for frame 5 when frames 7 and 8 arrive.) Should frame 6 then arrive perhaps it is late because it was lost the first time and had to he retransmitted, or perhaps it was simply delayed the receiver acknowledges frame 8, humps LFR to 8, and sets LAF to 12. If frame 6 was in fact lost, then a timeout will have occurred at the sender, causing it to retransmit frame 6.

We observe that when a timeout occurs, the amount of data in transit decreases, since the sender is unable to advance its window until frame 6 is acknowledged. This means that when packet losses occur, this scheme is no longer keeping the pipe full. The longer it takes to notice that a packet loss has occurred, the more severe this problem becomes.

Notice that in this example, the receiver could have sent a negative acknowledgment (NAK) for frame 6 as soon as frame 7 arrived. However, this is unnecessary since the sender's timeout mechanism is sufficient to catch this situation, and sending NAKs adds additional complexity to the receiver. Also, as we mentioned, it would have been legitimate to send additional acknowledgments of frame 5 when frames 7 and 8 arrived; in some cases, a sender can use duplicate ACKs as a clue that a frame was lost. Both approaches help to improve performance by allowing early detection of packet losses.

Yet another variation on this scheme would he to use selective acknowledgments. That is, the receiver could acknowledge exactly those frames it has received, rather than just the highest-numbered frame received in order. So, in the above example, the receiver could acknowledge the receipt of frames 7 and 8. Giving more information to the

sender makes it potentially easier for the sender to keep the pipe full, but adds complexity to the implementation.

The sending window size is selected according to how many frames we want to have outstanding on the link at a given time; SWS is easy to compute for a given delay x bandwidth product.' On the other hand, the receiver can set RWS to whatever it wants. TWO common settings are RWS = 1, which implies that the receiver will not buffer any frames that arrive out of order, and RWS = SWS, which implies that the receiver can buffer any of the frames the sender transmits. It makes no sense to set RWS > SWS since it's impossible for more than SWS frames to arrive out of order.

Finite Sequence Numbers and Sliding Window

We now return to the one simplification we introduced into the algorithm our assumption that sequence numbers can grow infinitely large. In practice, of course, a frame's sequence number is specified in a header field of some finite size. For example, a 3-bit field means that there are eight possible sequence numbers, 0. . . 7. This makes it necessary to reuse sequence numbers or, stated another way, sequence numbers wrap around. This introduces the problem of being able to distinguish between different incarnations of the same sequence numbers, which implies that the number of possible sequence numbers must he larger than the number of outstanding frames allowed. For example, stop-and-wait allowed one outstanding frame at a time and had two distinct sequence numbers.

Suppose we have one more number in our space of sequence numbers than we have potentially outstanding frames; that is, SWS MaxSeqNum - 1, where MaxSeqNum is the number of available sequence numbers. Is this sufficient? The answer depends on RWR. If RWS = 1, then MaxSeqNum SWS + 1 is sufficient. if RWS is equal to SWS, then having a MaxSeqNum just one greater than the sending 'window size is not good enough. To see this, consider the situation in which we have the eight sequence numbers 0 through 7, and SWS RWS 7. Suppose the sender transmits frames 0...6, they are successfully received, hut the ACKs are lost. The receiver is now expecting frames 7, 0. .5, but the sender times out and send frames 0...6. Unfortunately, the receiver is expecting the second incarnation of frames 0...5, hut gets the first incarnation of these frames. This is exactly the situation we wanted to avoid.

It turns out that the sending window size can be no more than half as big as the number of available sequence numbers when RWS = SWS, or stated more precisely,

SWS < (MaxSeqNum + 1)/2

Intuitively, what this is saying is that the sliding window protocol alternates between the two halves of the sequence number space, just as stop-and-wait alternates between sequence numbers 0 and 1. The only difference is that it continually slides between the two halves rather than discretely alternating between them.

Note that this rule is specific to the situation where RWS = SWS. We leave it as an exercise to determine the more general rule that works for arbitrary values of RWS and SWS. Also note that the relationship between the window size and the sequence number space depends on an assumption that is so obvious that it is easy to overlook, namely, that frames are not reordered in transit. This cannot happen on a direct point-to-point link since there is no way for one frame to overtake another during transmission.

1.8 Congestion

- When two many packets are present in the subnet, performance degrades. This situation is called Congestion. Figure below depicts the symptom.
- When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered, and the number delivered is proportional to the number sent.
- However, as traffic increases too far, the routers are no longer able to cope, and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely, and almost no packets are delivered.

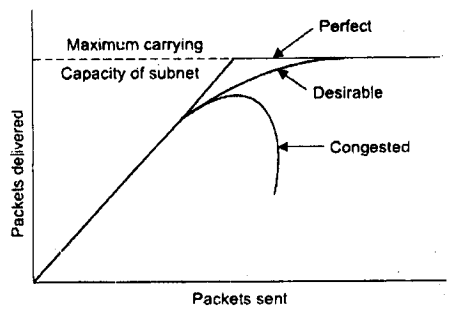


Figure 1.64 Figure Congestion

Congestion Control Principles

- Various metrics can be used to monitor the subnet for congestion. Chief among these are the percentage of all packets discarded for lack of buffer space, the average queue length, the number of packets that time out and are retransmitted, the average packet delay, and the standard deviation of packet delay.
- In all cases, rising numbers indicate growing congestion. The router that detects the congestion sends a packet to the traffic sources, announcing the problem.

Congestion Prevention Policies

Table below shows different data link, network, and transport policies that can affect congestion.

Table Policies that affect congestion

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

Leaky Bucket Algorithms

- Imagine a bucket with a small hole in the bottom is leaking at a constant rate, P, when there is any water in the bucket, and zero when the bucket is empty (Figure1.66 (a)). Also, once the bucket is full, any additional water entering it spills over the sides and is lost.

- The same idea can be applied to packets, as shown in Figure 1.65(*b*). Conceptually, each host is connected to the network by an interface containing a leaky bucket that is a finite internal queue.
- If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more processes within the host try to send a packet is unceremoniously discarded.
- This arrangement can be built into the hardware interface or simulated by the host operating system. It was first proposed by Turner and is called the **leaky bucket algorithm.**
- In fact, it is nothing other than a single-server queuing system with constant service time.

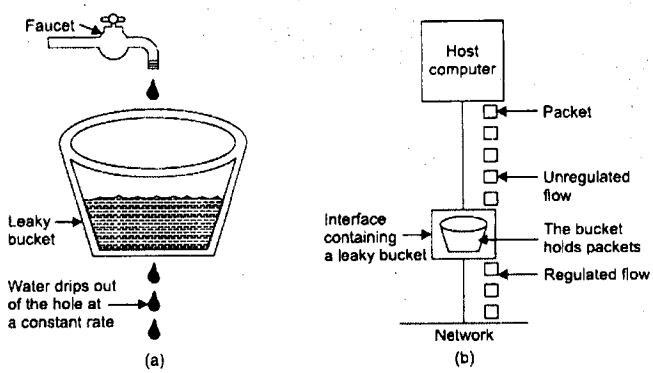


Figure 1.65 (a) A leaky bucket with water (b) A leaky bucket with packets

- The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user processes inside the host into an even flow of packets onto the network, smoothing out bursts and greatly reducing the chances of congestion.
- When the packets are all the same size, this algorithm can be used as described. However, when variable-sized packets are being used, it is often better to allow a fixed number of bytes per tick, rather than just one packet.
- Thus if the rule is 1024-bytes per tick, a single 1024-bytes packet can be admitted on a tick, two 512-byte packets, four 256-byte packets, and so on. If the residual byte count is too low, the next packet must wait until the next tick.

Implementation of Leaky Bucket

- The leaky bucket consists of a finite queue.
- When a packet arrives, if there is room on the queue it is appended to the queue, otherwise, it is discarded.
- At every clock tick, one packet is transmitted.
- The byte-counting leaky bucket is implemented almost the same way.
- At each tick, a counter is initialized to *n*.
- If the first packet on the queue has fewer bytes than the current value of the counter, it is transmitted, and the counter is decremented by that number of bytes.
- Additional packets may also be sent, as long as the counter is high enough.
- When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is overwritten and lost.

Token Bucket Algorithm

- The **leaky bucket algorithm** enforces a rigid output pattern at the average rate, no matter how bursty the traffic is.
- For many applications, it is better to allow the output to speed up somewhat when large bursts lose data. One such algorithm is the token bucket algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec.
- In Figure 1.66 (*a*) below we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In Figure 1.66 (*b*) below we see that three of the five packets have got through, but the other two are stuck waiting for two more tokens to be generated.

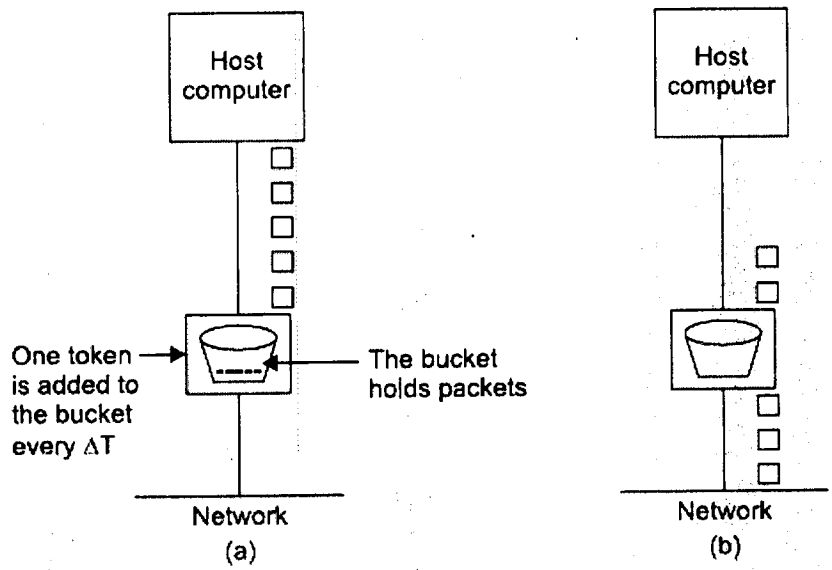


Figure 1.66 The token bucket algorithm (a) Before (b) After

- The **token bucket algorithm** provides a different king of traffic shaping than the leaky bucket algorithm does not allow idle hosts to save up permission to send large bursts later.
- The token bucket algorithm does allow saving, up to the maximum size of the bucket, *n*. This property means that bursts of up to *n* packets can be sent at one, allowing some burstiness in the output stream and giving faster response to sudden bursts of input.
- Another difference between the two algorithms is that the token bucket algorithm throws away tokens when the bucket fills up but never discards packets. In constant, the leaky bucket algorithm discards packets when the bucket fills up.

Choke Packets

- This is an approach that can be used for congestion in both virtual circuit and datagram subnets.
- Each router can easily monitor the utilization of its output lines and other resources. For example, its can associate with each line a real variable, μ whose value, between 0.0 and 1.0, reflects the recent utilization of that time.
- To maintain a good estimate of μ, a sample of the instantaneous line utilization, *f* (either 0 or 1), can be made periodically and μ updated according to

$\mu_{new} = a\mu_{old} + (1-a)f$

where, a determines how fast the router forest recent history.

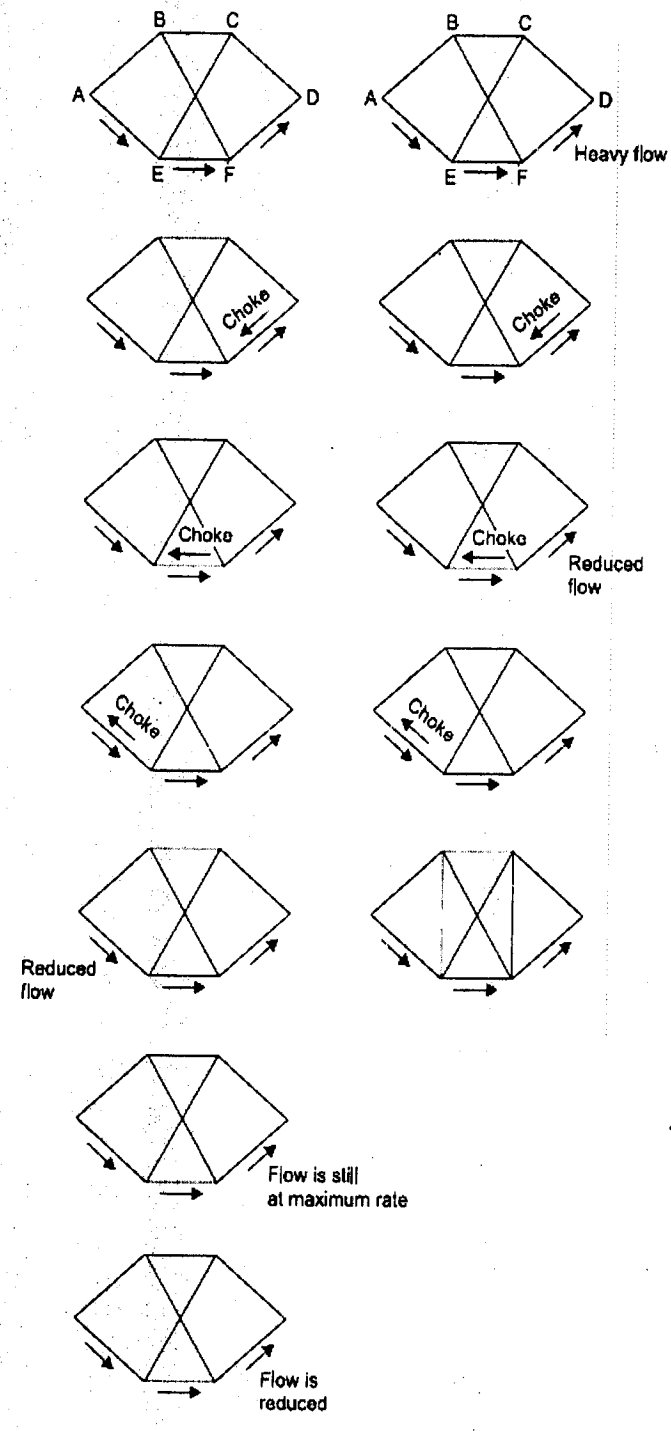
- Whenever μ moves above the threshold, the output line enters a "warning" state. Each newly arriving packet is checked to see if its output line is in warning state. If so, the router sends a **choke packet** back to the source host, giving it the destination found in the packet.
- The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets further along the path and is then forwarded in the usual way. When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by X per cent.
- Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval.
- After that period has expired, the host lists for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again.
- If no choke packets arrive during the listening period, the host may increase the flow again the feedback implicit in this protocol can help prevent congestion yet not throttle any flow unless trouble occurs.
- Hosts can reduce traffic by adjusting their policy parameters, *for example*, window size or leaky bucket output rate.
- Typically, the first choke packet causes the data rate to be reduced to 0.50 of its previous rate, the next one causes a reduction to 0.25, and so on. Increases are down in smaller increments to prevent congestion from reoccurring quickly.

Hop-by-Hop Choke Packets

- At high speeds and over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow.
- The 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.
- This is illustrated as shown in Figure below that congestion can be nipped in the bud without losing any packets. For example, a host in San Francisco (router A in Figure 1.67(*a*) that is sending traffic to a host in New York (router D in Figure (*a*) at 155 Mbps.)
- If the New York host begins to run out of buffers it will take about 30 msec for a choke packet to get back to San Francisco to tell it to slow down. The

choke packet propagation as the second, third, and fourth steps is given in Figure 1.67(*a*). In those 30 msec, another 4.6 megabits will have been sent.

- Even if the host in San Francisco completely shuts down immediately, the 4.6 megabits in the pipe will continue to put in and have to be dealt with. Only in the seventh diagram in Figure 1.67 (*a*) will the New York router



notice a slower flow.

Figure 1.67 (a) A choke packet that affects only the source

(b) A choke packet that affects each hop it passes through

Load Shedding

- When none of the above methods make the congestion disappear, routers can use 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 term comes from the world of electrical power generation where it refers to the practice of utilities intentionally blacking out certain area to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply. A router drowning in packets can just pick packets at random to drop, but usually it can do better than that. Which packet to discard may depend on the applications running. For file transfer, an old packet is worth more than a new one because dropping packet 6 and keeping packets 7 through 10 will cause a gap at the receiver that may force packets 6 through 10 to be retransmitted. In a 12 packet file, dropping 6 may require 7 through 12 to be retransmitted, whereas dropping 10 may require only 10 through 12 to be retransmitted. In contrast, for multimedia, a new packet is more important than an old one. The former policy is often called **win** and the latter is often called **milk**. A step above this in intelligence requires co-operation from the senders. For many applications, some packets are more important than others. For example, certain algorithms for compressing video periodically transmit and entire frame and then send subsequent frames as differences from the last full frame. In this case, dropping a packet that is part of a difference is preferable to dropping one that is part of a full frame. To implement an intelligent discard policy, applications must mark their packets in priority classes to indicate how important they are. If they do this, when packets have to be discarded, routers can first drop packets from the lowest class, then the next lowest class, and so on. Of course, unless there is some significant incentive to mark packets as anything other than **VERY IMPORTANT-NEVER. EVER DISCARD**, nobody will do it. The incentive might be in the form of money, this low-priority packets being cheaper to send than the high-priority ones. Alternatively, priority classes could be coupled with traffic shaping. Another option is to allow hosts to exceed the limits specified in the agreement negotiated when the virtual circuit was set up, but subject to the condition that all excess traffic be marked as low-priority. Such a strategy is actually not a bad idea, because it makes more efficient use of idle resources, allowing hosts to use them as long as nobody else is interested, but without

establishing a right to them when times get tough. Making packets by class requires one or more header bits in which to put the priority. ATM cells have 1-bit reserved in the header for this purpose, so every **ATM** cell is labeled either as low-priority or high-priority. **ATM** switches indeed use this bit when making discard decisions.

Jitter Control

- For applications such as audio and video transmission, it does not matter much if the packets take 20msec or 30msec to be delivered, as long as the transit time is constant. Having some packets taking 20msec and others taking 30msec will give an uneven quality to the sound or image. Thus the agreement might be that 99 per cent of the packets be delivered with a delay in the range of 24.5msec to 25.5 msec. The mean value chosen must be feasible, of course. In other words, an average amount of congestion must be calculated in. The jitter can be bounded by computing the expected transit time for each hop along the path. When a packet arrives at a router, the router checks to see how much the packet is behind or ahead of its schedule. This information is stored in the packet and updated at each hop. If the packet is ahead of schedule, it is held just long enough to get it back on schedule. If it is behind schedule, the router tries to get it out the door quickly. In fact, the algorithm for determining which of several packets competing for an output line should go next can always choose the packet furthers behind in its schedule. In this way, packets that are ahead of schedule get slowed down and packets that are behind schedule get speeded up, in both cases reducing the amount of jitter.

1.9 Multiplexing

In telecommunications and computer networks, **multiplexing** (also known as **muxing**) is a process where multiple analog message signals or digital data streams are combined into one signal over a shared medium. The aim is to share an expensive resource. For example, in telecommunications, several phone calls may be transferred using one wire. It originated in telegraphy, and is now widely applied in communications.

The multiplexed signal is transmitted over a communication channel, which may be a physical transmission medium. The multiplexing divides the capacity of the low-level communication channel into several higher-level logical channels, one for each message signal or data stream to be

transferred. A reverse process, known as demultiplexing, can extract the original channels on the receiver side.

A device that performs the multiplexing is called a multiplexer (MUX), and a device that performs the reverse process is called a demultiplexer (DEMUX).

History:

Time-division multiplexing was first developed in telegraphy; see multiplexing in telegraphy: Émile Baudot developed a time-multiplexing system of multiple Hughes machines in the 1870s.In 1962, engineers from Bell Labs developed the first D1 Channel Banks, which combined 24 digitised voice calls over a 4-wire copper trunk between Bell central office analogue switches. A channel bank sliced a 1.544Mbit/s digital signal into 8,000 separate frames, each composed of 24 contiguous bytes. Each byte represented a single telephone call encoded into a constant bit rate signal of 64 Kbit/s. Channel banks used a byte's fixed position (temporal alignment) in the frame to determine which call it belonged to.

Types of multiplexing

The group of multiplexing technologies may be divided into several types, all of which have significant variations:[1] space-division multiplexing (SDM), frequency-division multiplexing (FDM), time-division multiplexing (TDM), and code division multiplexing (CDM).

Variable bit rate digital bit streams may be transferred efficiently over a fixed bandwidth channel by means of statistical multiplexing, for example packet mode communication. Packet mode communication is an asynchronous mode time-domain multiplexing which resembles time-division multiplexing.

Digital bit streams can be transferred over an analog channel by means of code-division multiplexing (CDM) techniques such as frequency-hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS).

Space-division multiplexing

In wired communication, space-division multiplexing simply implies different point-to-point wires for different channels. One example is an analogue stereo audio cable, with one pair of wires for the left channel and another

for the right channel. Another example is a switched star network such as the analog telephone access network (although inside the telephone exchange or between the exchanges, other multiplexing techniques are typically employed) or a switched Ethernet network. A third example is a mesh network. Wired space-division multiplexing is typically not considered as multiplexing.

In wireless communication, space-division multiplexing is achieved by multiple antenna elements forming a phased array antenna. Examples are multiple-input and multiple-output (MIMO), single-input and multiple-output (SIMO) and multiple-input and single-output (MISO) multiplexing. For example, a IEEE 802.11n wireless router with *N* antennas makes it possible to communicate with *N* multiplexed channels, each with a peak bit rate of 54 Mbit/s, thus increasing the total peak bit rate with a factor *N*. Different antennas would give different multi-path propagation (echo) signatures, making it possible for digital signal processing techniques to separate different signals from each other. These techniques may also be utilized for space diversity (improved robustness to fading) or beam forming (improved selectivity) rather than multiplexing.

Frequency-division multiplexing:

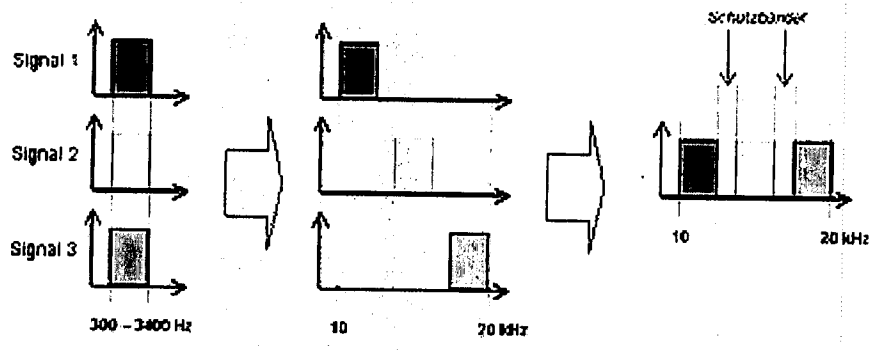


Figure 1.68 Frequency-division multiplexing

Frequency-division multiplexing (FDM): The spectrums of each input signal are shifted in several distinct frequency ranges.

Frequency-division multiplexing (FDM) is inherently an analog technology. FDM achieves the combining of several digital signals into one medium by sending signals in several distinct frequency ranges over that medium.

One of FDM's most common applications is cable television. Only one cable reaches a customer's home but the service provider can send multiple television channels or signals simultaneously over that cable to all subscribers. Receivers must tune to the appropriate frequency (channel) to access the desired signal.

Time-division multiplexing

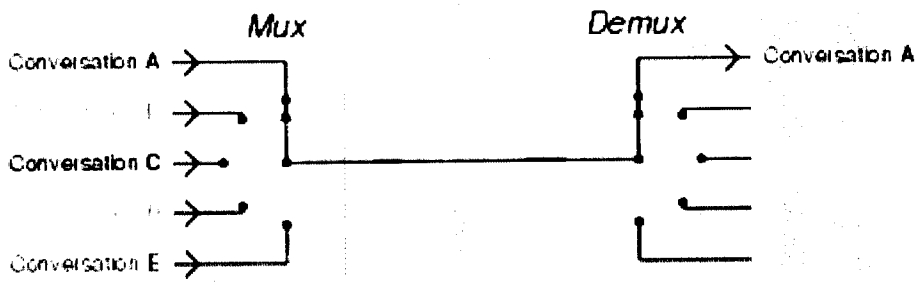


Figure 1.69 Time-division multiplexing (TDM).

Time-division multiplexing (TDM) is a digital technology. TDM involves sequencing groups of a few bits or bytes from each individual input stream, one after the other, and in such a way that they can be associated with the appropriate receiver. If done sufficiently and quickly, the receiving devices will not detect that some of the circuit time was used to serve another logical communication path.

Consider an application requiring four terminals at an airport to reach a central computer. Each terminal communicated at 2400 bps, so rather than acquire four individual circuits to carry such a low-speed transmission, the airline has installed a pair of multiplexers. A pair of 9600 bps modems and one dedicated analog communications circuit from the airport ticket desk back to the airline data center are also installed. Synchronous time division multiplexing (Sync TDM):

There are two types of Time-division multiplexing:

a. Synchronous Time division multiplexing (Sync TDM)

b. Statistical time-division multiplexing (Stat TDM)

There are three types of (Sync TDM): T1, SONET/SDH and ISDN

a. Synchronous digital hierarchy (SDH):

Plesiochronous digital hierarchy (PDH) was developed as a standard for multiplexing higher order frames. PDH created larger numbers of channels by multiplexing the standard Europeans 30 channel TDM frames. This solution worked for a while; however PDH suffered from several inherent drawbacks which ultimately resulted in the development of the Synchronous Digital Hierarchy (SDH). The requirements which drove the development of SDH were these: Be synchronous – All clocks in the system must align with a reference clock.

- Be service-oriented – SDH must route traffic from End Exchange to End Exchange without worrying about exchanges in between, where the bandwidth can be reserved at a fixed level for a fixed period of time.
- Allow frames of any size to be removed or inserted into an SDH frame of any size.
- Easily manageable with the capability of transferring management data across links.
- Provide high levels of recovery from faults.
- Provide high data rates by multiplexing any size frame, limited only by technology.
- Give reduced bit rate errors.

SDH has become the primary transmission protocol in most PSTN networks. It was developed to allow streams 1.544 Mbit/s and above to be multiplexed, in order to create larger SDH frames known as Synchronous Transport Modules (STM). The STM-1 frame consists of smaller streams that are multiplexed to create a 155.52 Mbit/s frame. SDH can also multiplex packet based frames e.g. Ethernet, PPP and ATM.

b. Statistical time-division multiplexing (Stat TDM):

STDM is an advanced version of TDM in which both the address of the terminal and the data itself are transmitted together for better routing. Using STDM allows bandwidth to be split over 1 line. Many college and corporate campuses use this type of TDM to logically distribute bandwidth. If there is one 10MBit line coming into the building, STDM can be used to provide 178

terminals with a dedicated 56k connection (178 * 56k = 9.96Mb). A more common use however is to only grant the bandwidth when that much is needed. STDM does not reserve a time slot for each terminal, rather it assigns a slot when the terminal is requiring data to be sent or received.

This is also called asynchronous time-division multiplexing(ATDM), in an alternative nomenclature in which "STDM" or "synchronous time division multiplexing" designates the older method that uses fixed time slots.

Code-division multiplexing :

Code division multiplexing (CDM) is a technique in which each channel transmits its bits as a coded channel-specific sequence of pulses. This coded transmission typically is accomplished by transmitting a unique time-dependent series of short pulses, which are placed within chip times within the larger bit time. All channels, each with a different code, can be transmitted on the same fiber and asynchronously demultiplexed. Other widely used multiple access techniques are Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA).

Code Division Multiplex techniques are used as an access technology, namely Code Division Multiple Access (CDMA), in Universal Mobile Telecommunications System (UMTS) standard for the third generation (3G) mobile communication identified by the ITU. Another important application of the CDMA is the Global Positioning System (GPS).

Relation to multiple access:

A multiplexing technique may be further extended into a multiple access method or channel access method, for example TDM into Time-division multiple access (TDMA) and statistical multiplexing into carrier sense multiple access (CSMA). A multiple access method makes it possible for several transmitters connected to the same physical medium to share its capacity.

Multiplexing is provided by the Physical Layer of the OSI model, while multiple access also involves a media access control protocol, which is part of the Data Link Layer.

The Transport layer in the OSI model as well as TCP/IP model provides statistical multiplexing of several application layer data flows to/from the same computer.

How codec used in multiplexing :

Enerdyne's **Codec 2X29** Video Compression and Multiplexing System represents the state of the art in video capture and sensor data transmission for applications such as UAVs or surveillance. It provides industry best flexibility in aggregation of sensor data into a single-bit stream in a package occupying only 29 cubic inches.

Features:

- Codec 2x29 features High-Quality, Low-Latency MJPEG Compression
- Allows tradeoff of frame rate, image quality, and output bit rate from 20 kbps to 20 Mbps
- NTSC or PAL, Composite, Y/C, or monochrome formats
- Multiplexing of multiple channels of synchronous data, asynchronous data, and toll quality digitized audio
- JPEG restart markers minimize error propagation effects

The **Codec 2X29** is the next generation design of Enerdyne's industry-leading ENC1000A29 (encoder) product, adding a wide range of multiplexing features to allow attachment of telemetry and sensor data to the compressed video stream with no requirements for synchronization of bit rates among different sensor data sources. The highly integrated Codec 2x29 design requires only a fraction of the power, volume and weight of previous codec and multiplexing solutions.

The Codec 2X29E is used as the encoder and multiplexer in a typical system configuration, while the Codec 2X29D is the decoder and demultiplexer. Both devices use the same Codec 2 circuit card to perform all processing functions

Application:

Telegraphy

The earliest communication technology using electrical wires, and therefore sharing an interest in the economies afforded by multiplexing, was the electric telegraph. Early experiments allowed two separate messages to

travel in opposite directions simultaneously, first using an electric battery at both ends, then at only one end.

- Émile Baudot developed a time-multiplexing system of multiple Hughes machines in the 1870s.
- In 1874, the quadruplex telegraph developed by Thomas Edison transmitted two messages in each direction simultaneously, for a total of four messages transiting the same wire at the same time.
- Several workers were investigating acoustic telegraphy, a frequency-division multiplexing technique, which led to the invention of the telephone.

Telephony

In telephony, a customer's telephone line now typically ends at the remote concentrator box down the street, where it is multiplexed along with other telephone lines for that neighborhood or other similar area. The multiplexed signal is then carried to the central switching office on significantly fewer wires and for much further distances than a customer's line can practically go. This is likewise also true for digital subscriber lines (DSL).

Fiber in the loop (FITL) is a common method of multiplexing, which uses optical fiber as the backbone. It not only connects POTS phone lines with the rest of the PSTN, but also replaces DSL by connecting directly to Ethernet wired into the home. Asynchronous Transfer Mode is often the communications protocol used. The concept is also now used in cable TV, which is increasingly offering the same services as telephone companies. IPTV also depends on multiplexing.

Video processing

In video editing and processing systems, multiplexing refers to the process of interleaving audio and video into one coherent MPEG transport stream (time-division multiplexing).

In digital video, such a transport stream is normally a feature of a container format which may include metadata and other information, such as subtitles. The audio and video streams may have variable bit rate. Software that produces such a transport stream and/or container is commonly called a statistical multiplexor or **muxer**. A **demuxer** is software that extracts or

otherwise makes available for separate processing the components of such a stream or container.

Digital broadcasting

In digital television and digital radio systems, several variable bit-rate data streams are multiplexed together to a fixed bitrates transport stream by means of statistical multiplexing. This makes it possible to transfer several video and audio channels simultaneously over the same frequency channel, together with various services.

In the digital television systems, this may involve several standard definition television (SDTV) programmes (particularly on DVB-T, DVB-S2, ISDB and ATSC-C), or one HDTV, possibly with a single SDTV companion channel over one 6 to 8 MHz-wide TV channel. The device that accomplishes this is called a statistical multiplexer. In several of these systems, the multiplexing results in an MPEG transport stream. The newer DVB standards DVB-S2 and DVB-T2 has the capacity to carry several HDTV channels in one multiplex. Even the original DVB standards can carry more HDTV channels in a multiplex if the most advanced MPEG-4 compressions hardware is used.

In digital radio, both the Eureka 147 system of digital audio broadcasting and the in-band on-channel HD Radio, FMeXtra, and Digital Radio Mondiale systems can multiplex channels. This is essentially required with DAB-type transmissions (where a multiplex is called an **ensemble**), but is entirely optional with IBOC systems.

Analog broadcasting

In FM broadcasting and other analog radio media, multiplexing is a term commonly given to the process of adding subcarriers to the audio signal before it enters the transmitter, where modulation occurs. Multiplexing in this sense is sometimes known as **MPX**, which in turn is also n old term for stereophonic FM, seen on stereo systems since the 1960s.

Advantage of multiplexing

- With advanced systems and more electronic features being added to modern cars, the need arises for a more reliable way to interconnect them.

- Instead of adding more wiring and more connections that may pose a reliability issue and add cost to the vehicle, manufactures can take advantage of multiplexing technology and send multiple sensor signals over fewer wires.
- Instead of adding more wiring, multiple sensors will send data to the vehicles computer over the same wire.
- Modern vehicles also incorporate networking technology where the vehicles computers may share information from a single sensor and use that information to perform a different task.
- For example a vehicles power train control module may use the input from a vehicle speed sensor to display the vehicles speed on the instrument panel.
- The power train control module may then share the vehicle speed information with the body controller to activate the door locks when the vehicle reaches a certain speed.

Disadvantage of multiplexing:

- Complex transmitters and receivers.
- They must be wide-band, which means they are more expensive and possibly less reliable.
- More complex circuitry required in each Module.
- There is potential reduction in the performance as certain events that share the same bus cannot take place in parallel.

1.10 The Telephone Network

The modern telephone network was developed to provide basic telephone service, which involves the two-way, real-time transmission of voice signals. In its most basic form, this service involves the transfer of an analog signal of a nominal bandwidth of 4 kHz across a sequence of transmission and switching facilities. The digital transmission capacity of this 4 kHz channel is about 45 kbps, which is miniscule in relation to the speed of modern computers. Nevertheless, the ubiquity and low cost of the telephone network make it an essential component of computer communications.

Telephone networks operate on the basis of circuit switching. Initially, circuit switching involved the setting up of a physical path from one telephone all the way across the network to the other telephone, as shown in Figure 1.70. At the telephone offices operators would make physical connections that would allow electric current to flow from one telephone to the other. The physical resources, such as wires and switch connections, were dedicated to the call for its entire duration. Modern digital telephone networks combine this circuit-switching approach to operating a network with digital transmission and digital switching.

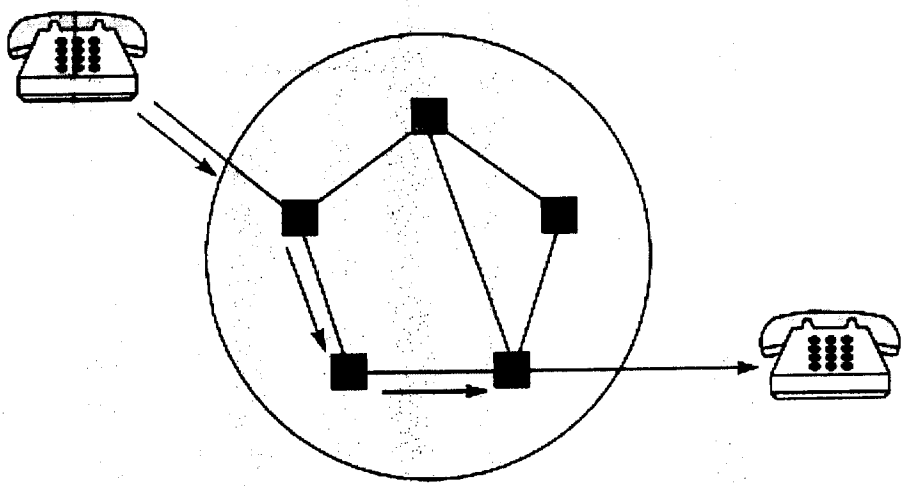


Figure 1.70 Circuit Switching

1.11 OSI Model for Networking

Concept of Layered Task

1. The main objective of a computer network is to be able to transfer the data from sender to receiver. This task can be done by breaking it into small sub tasks, each of which are well defined.

2. Each subtask will have its own process or processes to do and will take specific inputs and give specific outputs to the subtask before or after it. In more technical terms we can call these sub tasks as layers.

3. In general, every task or job can be done by dividing it into sub task or layers. Consider the example of sending a letter where the sender is in City A and receiver is in city B.

4. The process of sending letter is shown below:

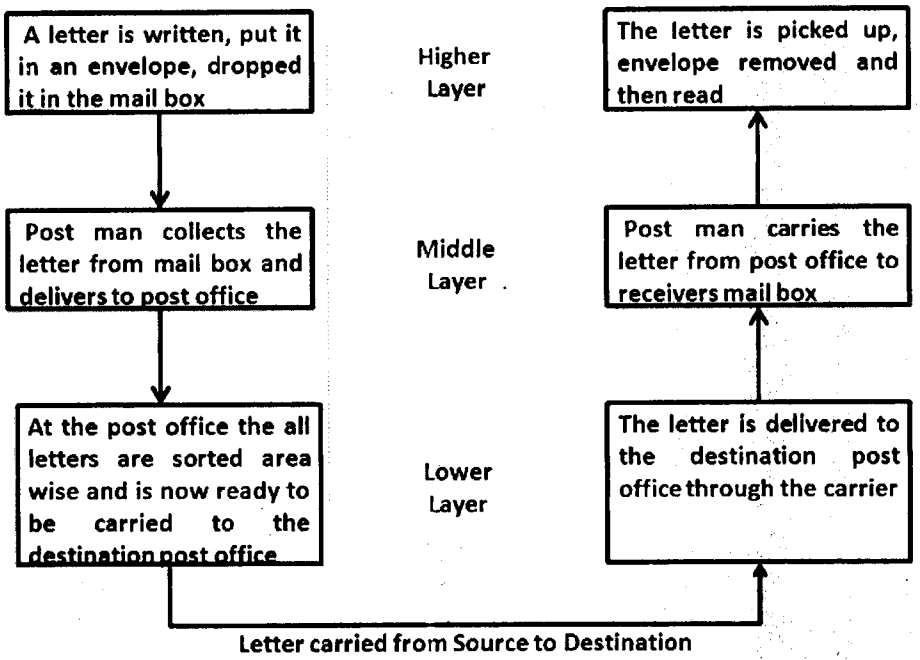


Figure 1.71 Concept of layer task: sending a letter

5. The above figure shows

a. Sender, Receiver & Carrier

b. Hierarchy of layers

6. At the sender site, the activities take place in the following descending order:

a. Higher Layer: The sender writes the letter along with the sender and receivers address and put it in an envelope and drop it in the mailbox.

b. Middle Layer: The letter is picked up by the post man and delivered to the post office

c. Lower Layer: The letters at the post office are sorted and are ready to be transported through a carrier.

7. During transition the letter may be carried by truck, plane or ship or a combination of transport modes before it reaches the destination post office.

8. At the Receiver site, the activities take place in the following ascending order:

a. Lower Layer: The carrier delivers the letter to the destination post office

b. Middle Layer: After sorting, the letter is delivered to the receivers mail box

c. Higher Layer: The receiver picks up the letter, opens the envelope and reads it.

9. Hierarchy of layers: The activities in the entire task are organized into three layers. Each activity at the sender or receiver side occurs in a particular order at the hierarchy.

10. The important and complex activities are organized into the Higher Layer and the simpler ones into middle and lower layer.

Open Systems Inter Connection Reference Model

Introduction to OSI Model & its layers

• The Open Systems Interconnection (OSI) Model was developed by International Organization for Standardization (ISO).

• ISO is the organization, OSI is the model

• It was developed to allow systems with different platforms to communicate with each other. Platform could mean hardware, software or operating system.

• It is a network model that defines the protocols for network communications.

• It is a hierarchical model that groups its processes into layers. It has 7 layers as follows: (Top to Bottom)

1. Application Layer

2. Presentation Layer

3. Session Layer

4. Transport Layer

5. Network Layer

6. Data Link Layer

7. Physical Layer

• Each layer has specific duties to perform and has to cooperate with the layers above and below it.

Layered Architecture of OSI Model

• The OSI model has 7 layers each with its own dedicated task.

• A message sent from Device A to Device B passes has to pass through all layers at A from top to bottom then all layers at B from bottom to top as shown in the figure below.

• At Device A, the message is sent from the top layer i.e Application Layer A then all the layers till it reaches its physical layer and then it is transmitted through the transmission medium.

• At Device B, the message received by the physical layer passes through all its other layers and moves upwards till it reaches its Application Layer.

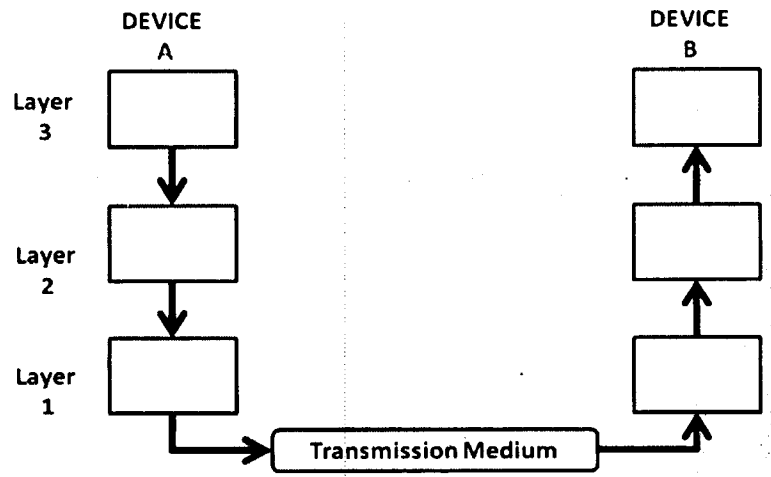


Figure 1.72 Flow of Data from Device A to Device B through various layers

• As the message travels from device A to device B, it may pass through many intermediate nodes. These intermediate nodes usually involve only the first three layers of the OSI model as shown below.

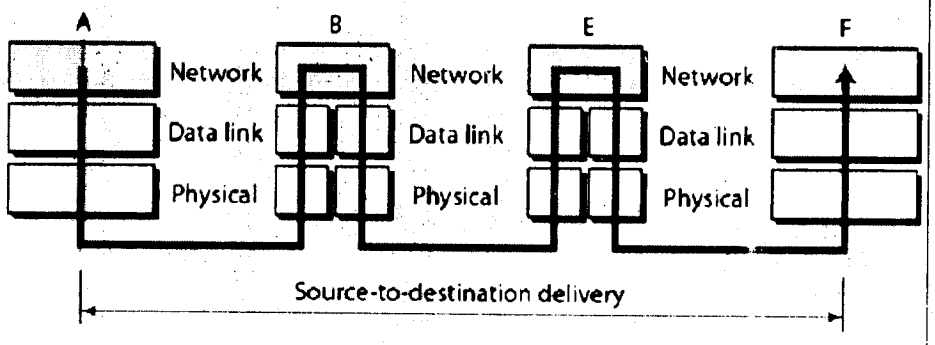


Figure 1.73 Data Transfer through Intermediate nodes

• The Data Link layer determines the next node where the message is supposed to be forwarded and the network layer determines the final recipient.

Communication & Interfaces

• For communication to occur, each layer in the sending device adds its own information to the message it receives from the layer just above it and passes the whole package to the layer just below it. Each layer in the receiving device removes the information added at the corresponding layer and sends the obtained data to the layer above it.

• Every Layer has its own dedicated function or services and is different from the function of the other layers.

• On every sending device, each layer calls upon the service offered by the layer below it.

• On every receiving device, each layer calls upon the service offered by the layer above it.

• Between two devices, the layers at corresponding levels communicate with each other .i.e layer 2 at receiving end can communicate and understand data from layer 2 of sending end. This is called peer - to - peer communication.

• For this communication to be possible between every two adjacent layers there is an interface. An interface defines the service that a layer must provide. Every layer has an interface to the layer above and below it as shown in the figure below

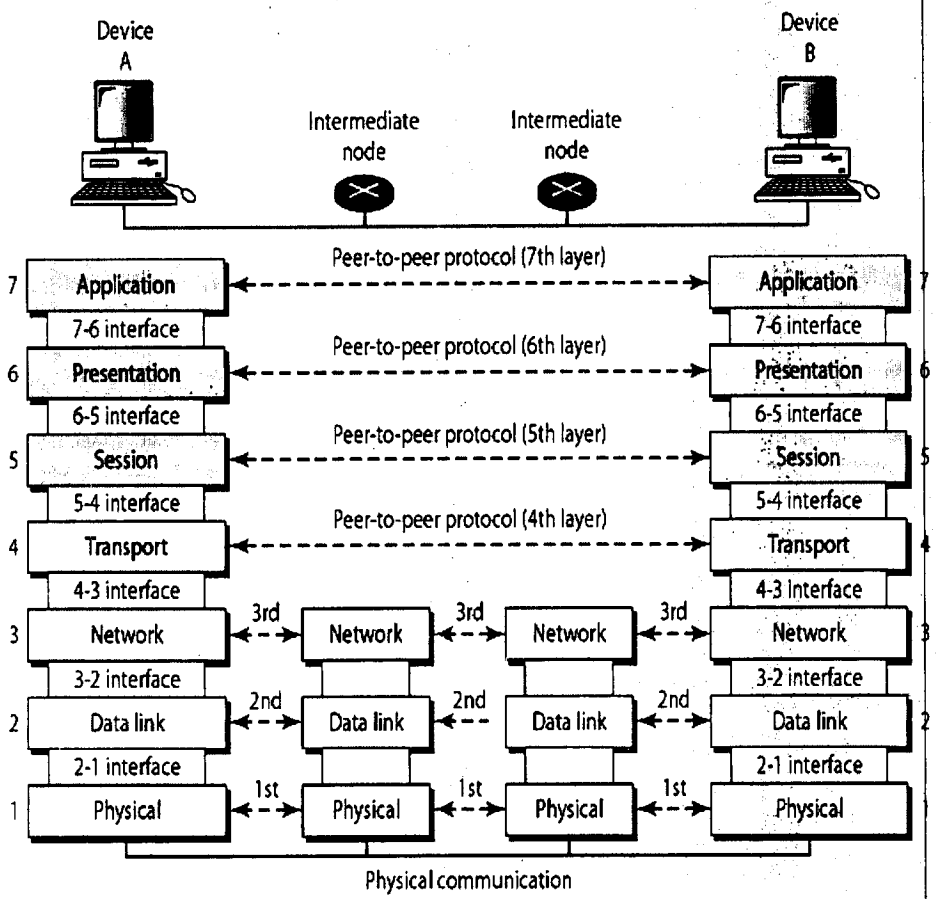


Figure 1.74 Communication & Interfaces in the OSI model

Encapsulation of Data

• As shown in the figure above the data at layer 7 i.e the Application layer along with the header added at layer 7 is given to layer 6, the Presentation layer. This layer adds Its header and passed the whole package to the layer below.

• The corresponding layers at the receiving side removes the corresponding header added at that layer and sends the remaining data to the above layer.

• The above process is called encapsulation

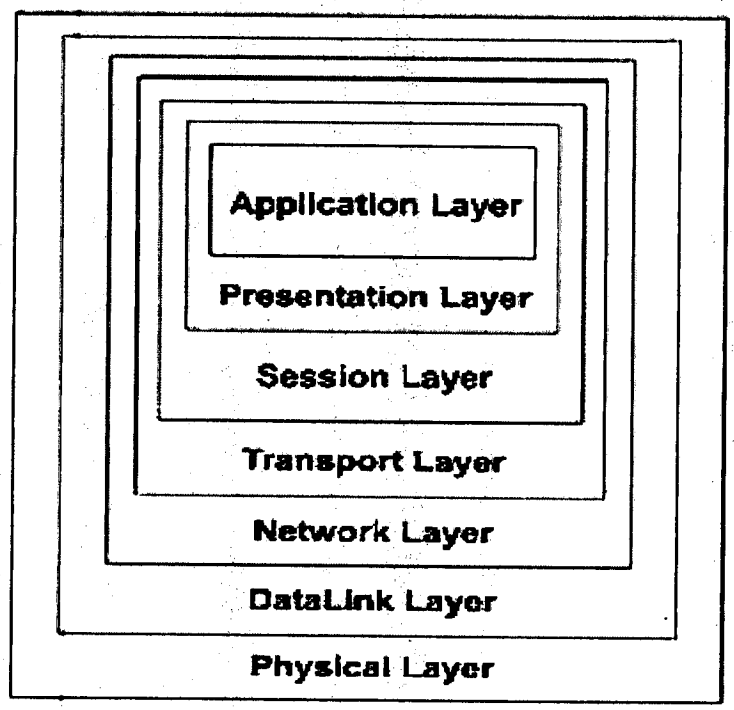


Figure 1.75 Encapsulation

Description of Layers in the OSI Model

Physical Layer

I. The Physical Layer provides a standardized interface to physical transmission media, including

a. Mechanical specification of electrical connectors and cables, for example maximum cable length

b. Electrical specification of transmission line

c. Bit-by-bit or symbol-by-symbol delivery

II. On the sender side, the physical layer receives the data from Data Link Layer and encodes it into signals to be transmitted onto the medium. On the receiver side, the physical layer receives the signals from the transmission medium decodes it back into data and sends it to the Data Link Layer as shown in the figure below:

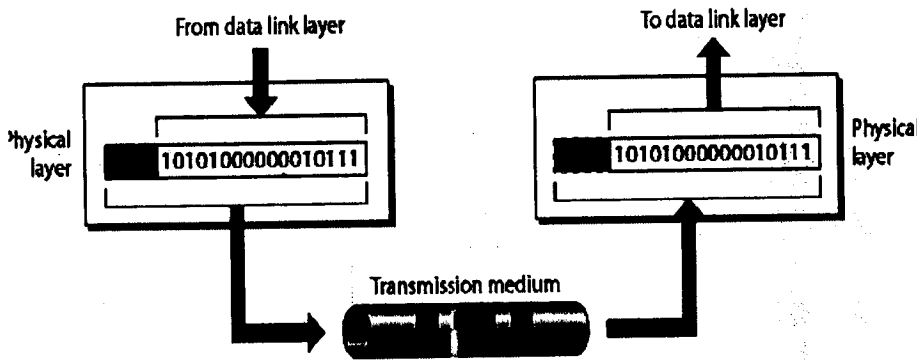


Figure 1.76 Transmission of data to and from Physical Layer

III. Interface: The Physical Layer defines the characteristics of interfaces between the devices & transmission medium.

IV. Representation of bits: The physical layer is concerned with transmission of signals from one device to another which involves converting data (l's & 0's) into signals and vice versa. It is not concerned with the meaning or interpretation of bits.

V. Data rate: The physical layer defines the data transmission rate i.e. number of bits sent per second. It is the responsibility of the physical layer to maintain the defined data rate.

VI. Synchronization of bits: To interpret correct and accurate data the sender and receiver have to maintain the same bit rate and also have synchronized clocks.

VII. Line configuration: The physical layer defines the nature of the connection .i.e. a point to point link, or a multi point link.

VIII. Physical Topology: The physical layer defines the type of topology in which the device is connected to the network. In a mesh topology it uses a multipoint connection and other topologies it uses a point to point connection to send data.

IX. Transmission mode: The physical layer defines the direction of data transfer between the sender and receiver. Two devices can transfer the data in simplex, half duplex or full duplex mode

X. Main responsibility of the physical layer: Transmission of bits from one hop to the next.

Data Link Layer

I. The Data Link layer adds reliability to the physical layer by providing error detection and correction mechanisms.

II. On the sender side, the Data Link layer receives the data from Network Layer and divides the stream of bits into fixed size manageable units called as Frames and sends it to the physical layer. On the receiver side, the data link layer receives the stream of bits from the physical layer and regroups them into frames and sends them to the Network layer. This process is called Framing. It is shown in the figure below:

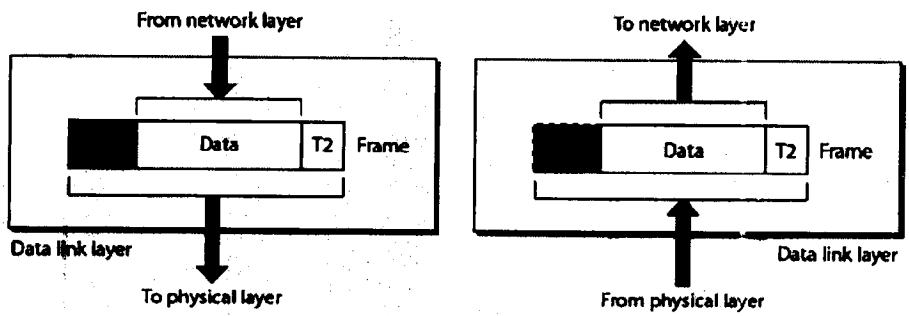


Figure 1.77 Data Link Layer: The process of Framing

III. Physical Addressing (inside I outside senders network)

a. The Data link layer appends the physical address in the header of the frame before sending it to physical layer.

b. The physical address contains the address of the sender and receiver.

c. In case the receiver happens to be on the same physical network as the sender; the receiver is at only one hop from the sender and the receiver address contains the receiver's physical address.

d. In case the receiver is not directly connected to the sender, the physical address is the address of the next node where the data is supposed to be delivered.

IV. Flow control

a. The data link layer makes sure that the sender sends the data at a speed at which the receiver can receive it else if there is an overflow at the receiver side the data will be lost.

b. The data link layer imposes flow control mechanism over the sender and receiver to avoid overwhelming of the receiver.

V. Error control

a. The data link layer imposes error control mechanism to identify lost or damaged frames, duplicate frames and then retransmit them.

b. Error control information is present in the trailer of a frame.

VI. Access Control: The data link layer imposes access control mechanism to determine which device has right to send data in a multipoint connection scenario.

VII. Main Responsibility: The main responsibility of the data link layer is hop to hop transmission of frames.

Network Layer

I. The network layer makes sure that the data is delivered to the receiver despite multiple intermediate devices.

II. The network layer at the sending side accepts data from the transport layer, divides it into packets, adds addressing information in the header and passes it to the data link layer. At the receiving end the network layer receives the frames sent by data link layer, converts them back into packets, verifies the physical address (verifies if the receiver address matches with its own address) and the send the packets to the transport layer.

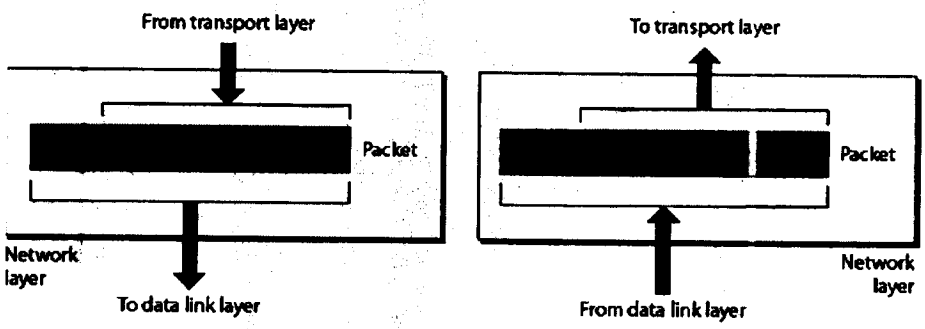


Figure 1.78 Network Layer

III. The network layer is responsible for source to destination of delivery of data. Hence it may have to route the data through multiple networks via multiple intermediate devices. In order to achieve this network layer relies on two things:

a. Logical Addressing

b. Routing

IV. Logical Addressing

• The network layer uses logical address commonly known as IP address to recognize devices on the network.

• An IP address is a universally unique address which enables the network layer to identify devices outside the sender's network.

• The header appended by the network layer contains the actual sender and receiver IP address.

• At every hop the network layer of the intermediate node check the IP address in the header, if its own IP address does not match with the IP address of the receiver found in the header, the intermediate node concludes that it is not the final node but an intermediate node and passes the packet to the data link layer where the data is forwarded to the next node.

V. Routing

•. The network layer divides data into units called packets of equal size and bears a sequence number for rearranging on the receiving end.

• Each packet is independent of the other and may travel using different routes to reach the receiver hence may arrive out of turn at the receiver.

• Hence every intermediate node which encounters a packet tries to compute the best possible path for the packet. The best possible path may depend on several factors such as congestion, number of hops, etc

• This process of finding the best path is called as Routing. It is done using routing algorithms.

VI. Main Responsibility: The main responsibility of Network Layer is transmission of packets from source to destination.

Transport Layer

I. A logical address at network layer facilitates the transmission of data from source to destination device. But the source and the destination both may be having multiple processes communicating with each other. Hence it is important to deliver the data not only from the sender to the receiver but from the correct process on the sender to the correct process on the receiver. The transport layer takes care of process to process delivery of data and makes sure that it is intact and in order.

II. At the sending side, the transport layer receives data from the session layer, divides it into units called segments and sends to the network layer. At the receiving side, the transport layer receives packets from the network layer, converts and arranges into proper sequence of segments and sends it to the session layer.

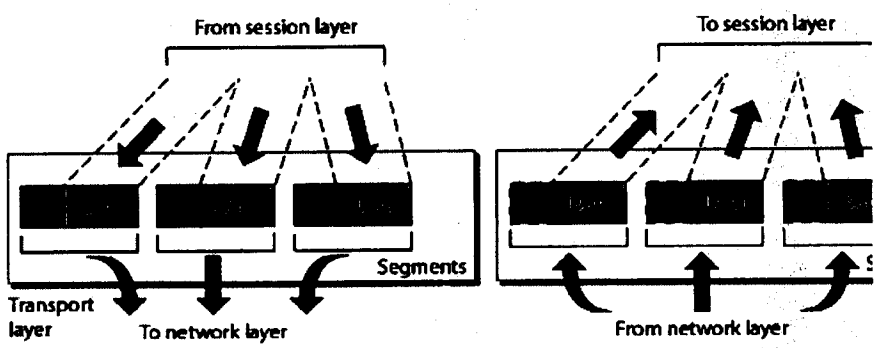


Figure 1.79 Transport Layer

III. To ensure process to process delivery the transport layer makes use of port address to identify the data from the sending and receiving process. A Port Address is the name or label given to a process. It is a 16 bit address. Ex. TELNET uses port address 23, HTTP uses port address 80. Port address is also called as Service Point Address

IV. The data can be transported in a connection oriented or connectionless manner. If the connection is connection oriented then all segments are received in order else they are independent of each other and are received out of order and have to be rearranged.

V. The Transport layer is responsible for segmentation and reassembly of the message into segments which bear sequence numbers. This numbering enables the receiving transport layer to rearrange the segments in proper order.

VI. Flow Control & Error control: the transport layer also carries out flow control and error control functions; but unlike data link layer these are end to end rather than node to node.

VII. Main Responsibility: The main responsibility of the transport layer is process to process delivery of the entire message.

Session Layer

I. The session layer establishes a session between the communicating devices called dialog and synchronizes their interaction. It is the responsibility of the session layer to establish and synchronize the dialogs. It is also called the network dialog controller.

II. The session layer at the sending side accepts data from the presentation layer adds checkpoints to it called syn bits and passes the data to the transport layer. At the receiving end the session layer receives data from the transport layer removes the checkpoints inserted previously and passes the data to the presentation layer.

III. The checkpoints or synchronization points is a way of informing the status of the data transfer. Ex. A checkpoint after first 500 bits of data will ensure that those 500 bits are not sent again in case of retransmission at 65O bit.

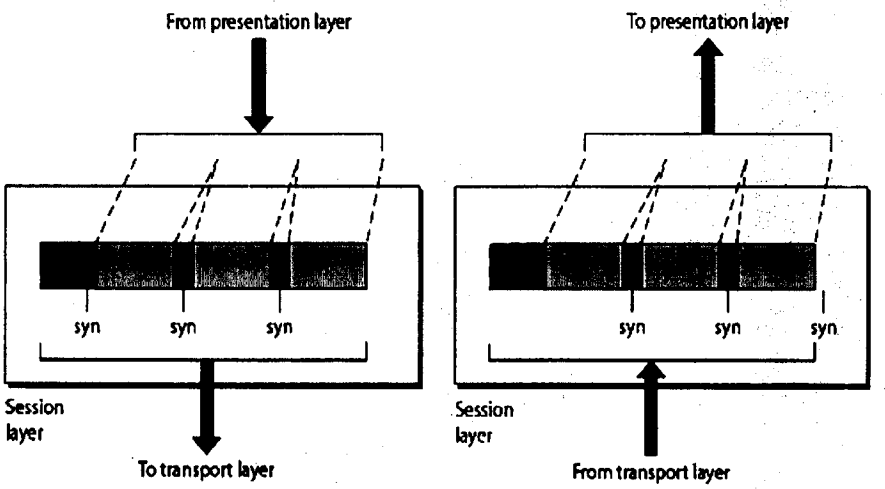


Figure 1.80 Session layer

Presentation Layer

I. The communicating devices may be having different platforms. The presentation layer performs translation, encryption and compression of data.

II. The presentation layer at sending side receives the data from the application layer adds header which contains information related to encryption and compression and sends it to the session layer. At the receiving side, the presentation layer receives data from the session layer decompresses and decrypts the data as required and translates it back as per the encoding scheme used at the receiver.

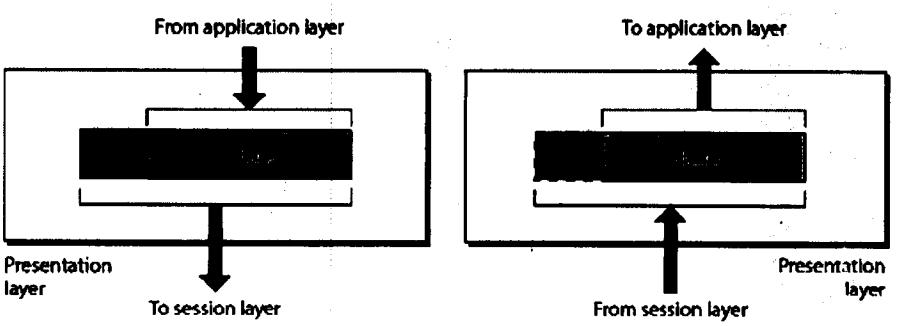


Figure 1.81 Presentation layer

III. Translation: The sending and receiving devices may run on different platforms (hardware, software and operating system). Hence it is important

that they understand the messages that are used for communicating. Hence a translation service may be required which is provided by the Presentation layers

IV. Compression: Compression ensures faster data transfer. The data compressed at sender has to be decompressed at the receiving end, both performed by the Presentation layer.

V. Encryption: It is the process of transforming the original message to change its meaning before sending it. The reverse process called decryption has to be performed at the receiving end to recover the original message from the encrypted message.

VI. Main responsibility: The main responsibility of the Presentation layer is translation, compression and encryption.

Application Layer

I. The application layer enables the user to communicate its data to the receiver by providing certain services. For ex. Email is sent using X.400 service.

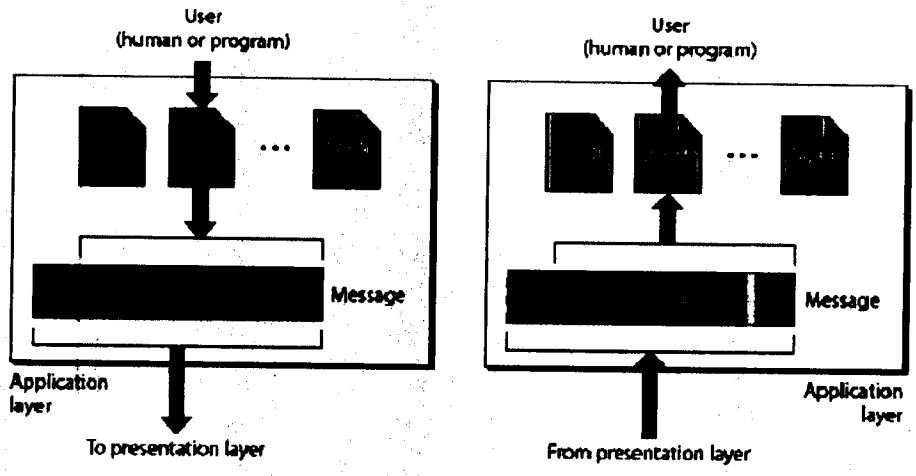


Figure 1.82 Application Layer

II. X500 is a directory service used to provide information and access to distributed objects

III. X400 is services that provides basis for mail storage and forwarding

IV. FTAM (File transfer, access and management) provides access to files stored on remote computers and mechanism for transfer and manage them locally.

V. Main Responsibility: Main Responsibility of Application layer is to provide access to network resources.

1.12 The Internet

The Internet has revolutionized many aspects of our daily lives. It has affected the way we do business as well as the way we spend our leisure time. Count the ways you've used the Internet recently. Perhaps you've sent electronic mail (e-mail) to a business associate, paid a utility bill, read a newspaper from a distant city, or looked up a local movie schedule-all by using the Internet. Or maybe you researched a medical topic, booked a hotel reservation, chatted with a fellow Trekkie, or comparison-shopped for a car. The Internet is a communication system that has brought a wealth of information to our fingertips and organized it for our use.

The Internet is a structured, organized system. We begin with a brief history of the Internet. We follow with a description of the Internet today.

A Brief History

A network is a group of connected communicating devices such as computers and printers. An internet (note the lowercase letter i) is two or more networks that can communicate with each other. The most notable internet is called the Internet (uppercase letter I), a collaboration of more than hundreds of thousands of interconnected networks. Private individuals as well as various organizations such as government agencies, schools, research facilities, corporations, and libraries in more than 100 countries use the Internet. Millions of people are users. Yet this extraordinary communication system only came into being in 1969.

In the mid-1960s, mainframe computers in research organizations were standalone devices. Computers from different manufacturers were unable to communicate with one another. The Advanced Research Projects Agency (ARPA) in the Department of Defense (DoD) was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for ARPANET, a small network of connected computers. The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an interface message processor (IMP). The IMPs, in turn, would be connected to one another. 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. Four nodes, at the University of California at Los Angeles (UCLA), the University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI), and the University of Utah, were connected via the IMPs to form a network. Software called the Network control Protocol (NCP) provided communication between the hosts.

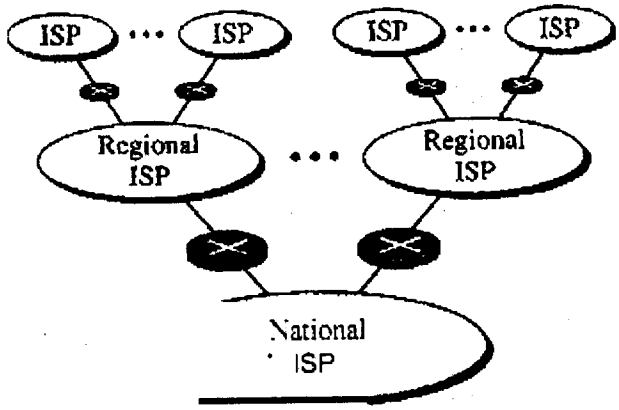
In 1972, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the Intern citing Project. Cerf and Kahn's landmark 1973 paper outlined the protocols to achieve end-to-end delivery of packets. This paper on Transmission Control Protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway.

Shortly thereafter, authorities made a decision to split TCP into two protocols: Transmission Control Protocol (TCP) and Internetworking Protocol (IP). IP would handle datagram muting while TCP would be responsible for higher-level functions such as segmentation, reassembly, and error detection. The internetworking protocol became known as TCP/IP.

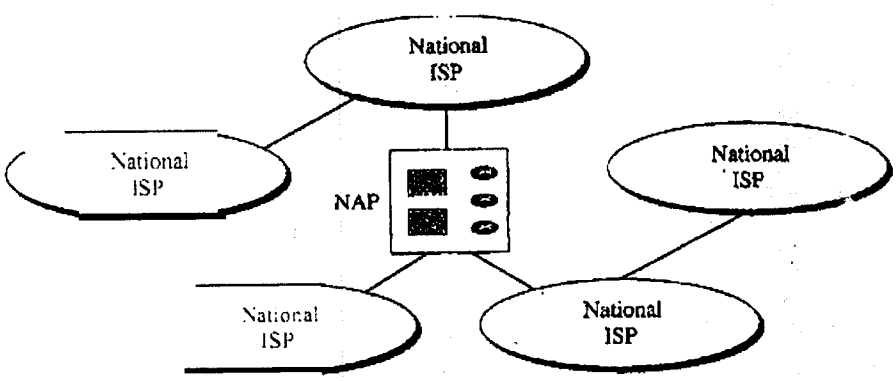
The Internet Today

The Internet has come a long way since the 1960s. The Internet today is not a simple hierarchical structure. It is made up of many wide- and local-area networks joined by connecting devices and switching stations. It is difficult to give an accurate it presentation of the Internet because it is continually changing-new networks are being added, existing networks are adding addresses, and networks of defunct companies are being removed. Today most end users who want Internet connection use the services of Internet service providers (ISPs). There are international service providers, national service providers, regional service providers, and local service providers. The Internet today is nm by private companies, not the

government. Figure 1.83 shows a conceptual (not geographic) view of the Internet.



a. Structure of a national ISP



b. Interconnection of national ISPs

Figure 1.83 Hierarchical organization of the Internet

International Internet Service Providers

At the top of the hierarchy are the international service providers that connect nations together.

National Internet Service Providers

The national Internet service providers are backbone networks created and maintained by specialized companies. There are many national ISPs operating in North America; some of the most well known are SprintLink, PSINet, UUNet Technology, AGIS, and internet Mel. To provide

connectivity between the end users, these backbone networks are connected by complex switching stations (normally run by a third party) called network access points (NAPs). Some national ISP networks are also connected to one another by private switching stations called peering points. These normally operate at a high data rate (up to 600 Mbps).

Regional In met Service Providers

Regional Internet service providers or regional ISPs are smaller ISPs that are connected to one or more national ISPs. They are at the third level of the hierarchy with a smaller data rate.

Local Internet Service Provides

Local Internet service providers provide direct service to the end users. The local ISPs can be connected to regional ISPs or directly to national ISPs. Most end users are connected to the local ISPs. Note that in this sense, a local ISP can be a company that just provides Internet services, a corporation with a network that supplies services to its own employees, or a nonprofit organization, such as a college or a university that runs its own network. Each of these local ISPs can be connected to a regional or national service provider.

1.13 Asynchronous Transfer Mode (ATM)

Asynchronous Transfer Mode (ATM) is the cell relay protocol designed by the ATM Forum and adopted by the ITU-T. The combination of ATM and SONET will allow high-speed interconnection of all the world's networks. In fact, ATM can be thought of as the "highway" of the information superhighway.

Design Goals

Among the challenges faced by the designers of ATM, six stand out.

1. Foremost is the need for a transmission system to optimize the use of high-data-rate transmission media, in particular optical fiber. In addition to offering large bandwidths, newer transmission media and equipment are dramatically less susceptible to noise degradation. A technology is needed to take advantage of both factors and thereby maximize data rates.

2. The system must interface with existing systems and provide wide-area interconnectivity between them without lowering their effectiveness or requiring their replacement.

3. The design must be implemented inexpensively so that cost would not be a barrier to adoption. If ATM is to become the backbone of international communications, as intended, it must be available at low cost to every user who wants it.

4. The new system must be able to work with and support the existing telecommunications hierarchies (local loops, local providers, long-distance carriers, and so on).

5. The new system must be connection-oriented to ensure accurate and predictable delivery.

6. Last but not least, one objective is to move as many of the functions to hardware as possible (for speed) and eliminate as many software functions as possible (again for speed).

Prob1ers

Before we discuss the solutions to these design requirements, it is useful to examine some of the problems associated with existing systems.

Frame Networks

Before ATM, data communications at the data link layer had been based on frame switching and frame networks. Different protocols use frames of varying size and intricacy. As networks become more complex, the information that must be carried in the header becomes more extensive. The result is larger and larger headers relative to the size of the data unit. In response, some protocols have enlarged the size of the data unit to make header use more efficient (sending more data with the same size header). Unfortunately, large data fields create waste. If there is not much information to transmit, much of the field goes unused. To improve utilization, some protocols provide variable frame sizes to users.

Mixed Aetwork Traffic

As you can imagine, the variety of frame sizes makes traffic unpredictable. Switches, multiplexers, and routers must incorporate elaborate software

systems to manage the various sizes of frames. A great deal of header information must be read, and each bit counted and evaluated to ensure the integrity of every frame. Internetworking among the different frame networks is slow and expensive at best, and impossible at worst,

Another problem is that of providing consistent data rate delivery when frame sizes are unpredictable and can vary so dramatically. To get the most out of broadband technology, traffic must be time-division multiplexed onto shared paths. Imagine the results of multiplexing frames from two networks with different requirements (and frame designs) onto one link (see Figure 1.84). What happens when line I uses large frames (usually data frames) while line 2 uses very small frames (the norm for audio and video information)?

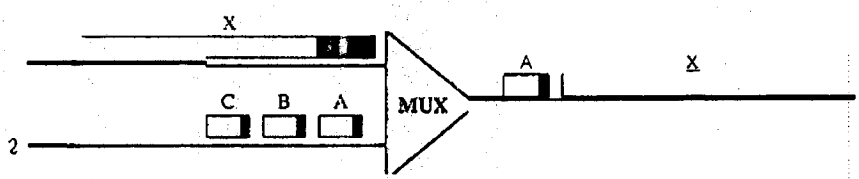


Figure 1.84 Multiplexing using different frame sizes

If line I's gigantic frame X arrives at the multiplexer even a moment earlier than line 2's frames, the multiplexer puts frame X onto the new path first. After all, even if line 2's frames have priority, the multiplexer has no way of knowing to wait for them and so processes the frame that has arrived. Frame A must therefore wait for the entire X bit stream to move into place before it can follow. The sheer size of X creates an unfair delay for frame A. The same imbalance can affect all the frames from line 2.

Because audio and video frames ordinarily are small, mixing them with conventional data traffic often creates unacceptable delays of this type and makes shared frame links unusable for audio and video information. Traffic must travel over different paths. in much the same way that automobile and train traffic does. But to fully utilize broad bandwidth links, we need to be able to send all kinds of traffic over the same links.

Cell Networks

Many of the problems associated with frame internetworking are solved by adopting a concept called cell networking. A cell is a small data unit of fixed size. In a cell network, which uses the cell as the basic unit of data exchange, all data are loaded into identical cells that can be transmitted

with complete predictability and uniformity. As frames of different sizes and formats reach the cell network from a tributary network, they are split into multiple small data units of equal length and are loaded into cells. The cells are then multiplexed with other cells and routed through the cell network. Because each cell is the same size and all are small, the problems associated with multiplexing different-sized frames are avoided.

Figure 1.85 shows the multiplexer from Figure 1.84 with the two lines sending cells instead of frames. Frame X has been segmented into three cells: X, Y, and Z. Only the first cell from line I gets put on the link before the first cell from line 2. The cells from the two lines are interleaved so that none suffers a long delay.

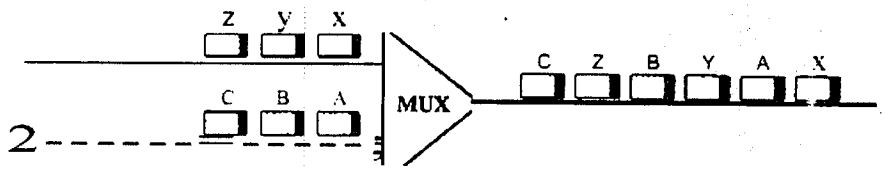


Figure 1.85 Multiplexing using cells

A second point in this same scenario is that the high speed of the links coupled with the small size of the cells means that, despite interleaving, cells from each line arrive at their tvspective destinations in an approximation of a continuous stream (much as a movie appears to your brain to be continuous action when in fact it is really a series of separate. still photographs). In this way, a cell network can handle mal-time transmissions, such as a phone call, without the parties being aware of the segmentation or multiplexing at all.

Asynchronous TDM

ATM uses asynchronous time-division multiplexing-that is why it is called Asynchronous Transfer Mode-to multiplex cells corning from different channels. It uses fixed-size slots (size of a cell). ATM multiplexers fill a slot with a cell from any input channel that has a cell; the slot is empty if none of the channels has a cell to send.

Figure 1.86 shows how cells from three inputs are multiplexed. At the first tick of the clock: channel 2 has no cell (empty input slot), so the multiplexer fills the slot with a cell from the third channel. When all the cells from all the channels are multiplexed, the output slots are empty.

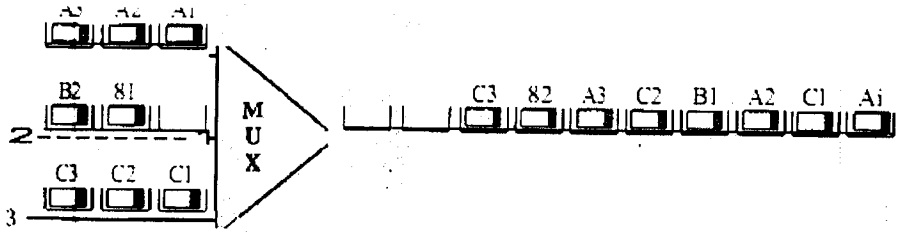


Figure 1.86 ATM multiplexing

Architecture

ATM is a cell-switched network. The user access devices, called the endpoints, are connected through a user-to-network interface (UNI) to the switches inside the network. The switches are connected through network-to-network interfaces (NNIs). Figure 1.87 shows an example of an ATM network.

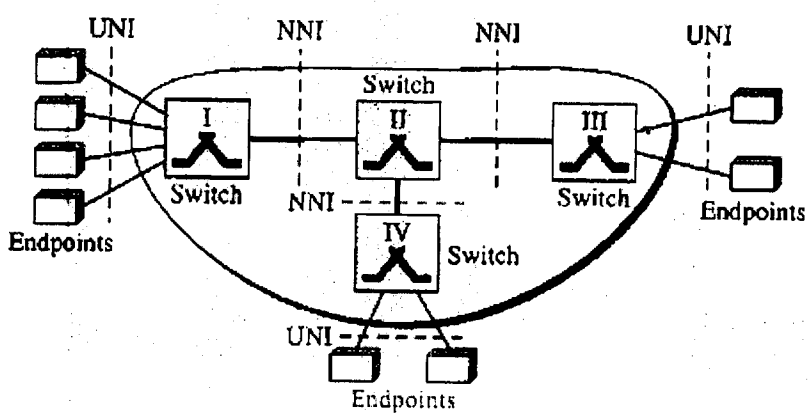


Figure 1.87 Architecture of an ATM network

Virtual Connection

Connection between two endpoints is accomplished through transmission paths (TPs), virtual paths (VPs), and virtual circuits (VCs). A transmission path (TP) is the physical connection (wire, cable, satellite, and so on) between an endpoint and a switch or between two switches. Think of two switches as two cities. A transmission path is the set of all highways that directly connect the two cities.

A transmission path is divided into several virtual paths. A virtual path (VP) provides a connection or a set of connections between two switches. Think

of a virtual path as a highway that connects two cities. Each highway is a virtual path; the set of all highways is the transmission path.

Cell networks are based on virtual circuits (VCs). All cells belonging to a single message follow the same virtual circuit and remain in their original order until they reach their destination. Think of a virtual circuit as the lanes of a highway (virtual path). Figure 1.88 shows the relationship between a transmission path (a physical connection), virtual paths (a combination of virtual circuits that are bundled together because parts of their paths are the same), and virtual circuits that logically connect two points.

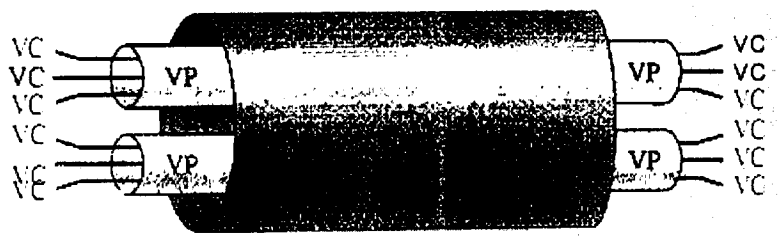


Figure 1.88 TP, VPs, and VCs

To better understand the concept of VPs and VCs, look at Figure 1.89. In this figure, eight endpoints are communicating using four VCs. However, the first two VCs seem to share the same virtual path from switch I to switch III, so it is reasonable to bundle these two VCs together to form one VP. On the other hand, it is clear that the other two VCs share the same path from switch Ito switch IV, so it is also reasonable to combine them to form one VP.

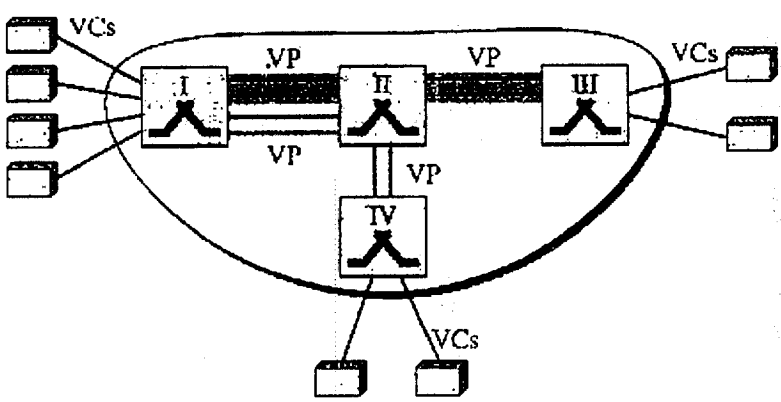


Figure 1.89 Example of VPs and VCs

Identifiers In a virtual circuit network, to route data from one endpoint to another, the virtual connections need to be identified. For this purpose, the designers of ATM created a hierarchical identifier with two levels: a virtual path identifier (VPI) and a virtual-circuit identifier (VCI). The VPI defines the specific VP, and the VCI defines a particular VC inside the VP. The VPI is the same for all virtual connections that are bundled (logically) into one VP.

Cells

The basic data unit in an ATM network is called a cell. A cell is only 53 bytes long with 5 bytes allocated to the header and 48 bytes carrying the payload (user data may be less than 48 bytes). We will study in detail the fields of a cell, but for the moment it suffices to say that most of the header is occupied by the VPI and VCI that define the virtual connection through which a cell should travel from an endpoint to a switch or from a switch to another switch. Figure 1.90 shows the cell structure.

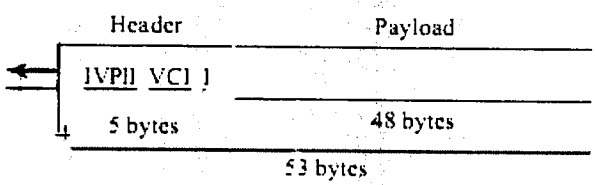
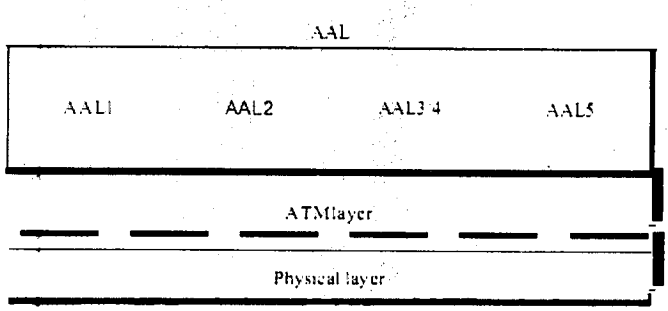


Figure 1.90 An ATM cell

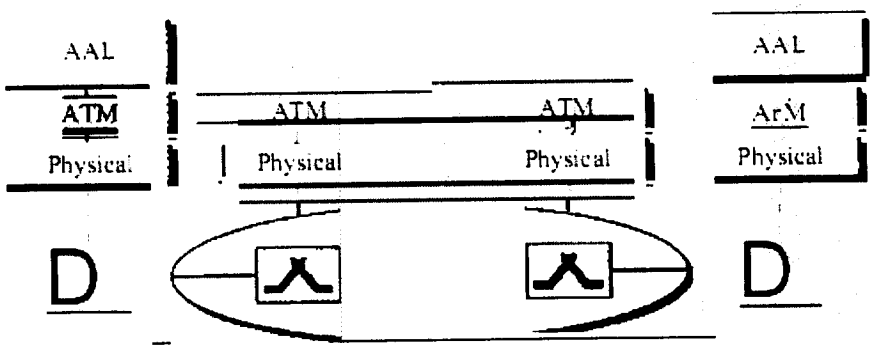
ATM Layers

The ATM standard defines three layers. They are, from top to bottom, the application adaptation layer, the ATM layer, and the physical layer.

The endpoints use all three layers while the switches use only the two bottom layers (see Figure 1.91).



a. ATM layers



b. ATM layers in endpoint devices and switches

Figure 1.91 ATM layers

Physical Layer

Like Ethernet and wireless LANs, ATM cells can be carried by any physical layer carrier.

SONET The original design of ATM was based on SONET as the physical layer carrier. SONET is preferred for two reasons. First, the high data rate of SONET's carrier reflects the design and philosophy of ATM. Second, in using SONET, the boundaries of cells can be clearly defined. SONET specifies the use of a pointer to define the beginning of a payload. If the beginning of the first ATM cell is defined, the rest of the cells in the same payload can easily be identified because there are no gaps between cells. Just count 53 bytes ahead to find the next cell.

Other Physical Technologies ATM does not limit the physical layer to SONET. Other technologies, even wireless, may be used. However, the problem of cell boundaries must be solved. One solution is for the receiver to guess the end of the cell and apply the CRC to the 5-byte header. If there is no error, the end of the cells found, with a high probability, correctly. Count 52 bytes back to find the beginning of the cell.

ATM Layer

The ATM layer provides routing, traffic management. Switching, and multiplexing services. It processes outgoing traffic by accepting 48-byte segments from the AAL sub layers and transforming them into 53-byte cells by the addition of a 5-byte header (see Figure 1.92).

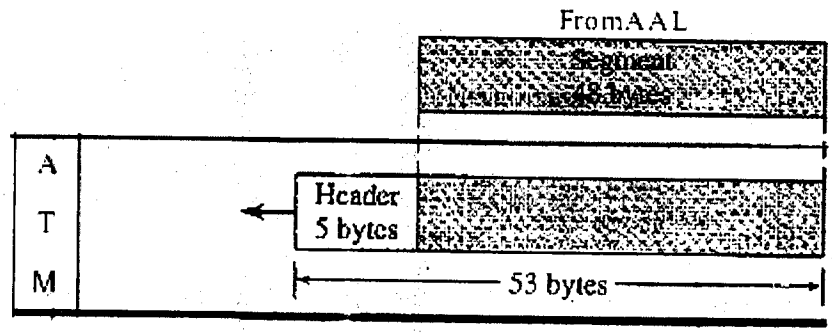


Figure 1.92 ATM layer

Header Format ATM uses two formats for this header, one for user-to-network interface (UNI) cells and another for network-to-network interface (NNI) cells, Figure 1.93 shows these headers in the byte-by-byte format preferred by the ITU-T (each row represents a byte).

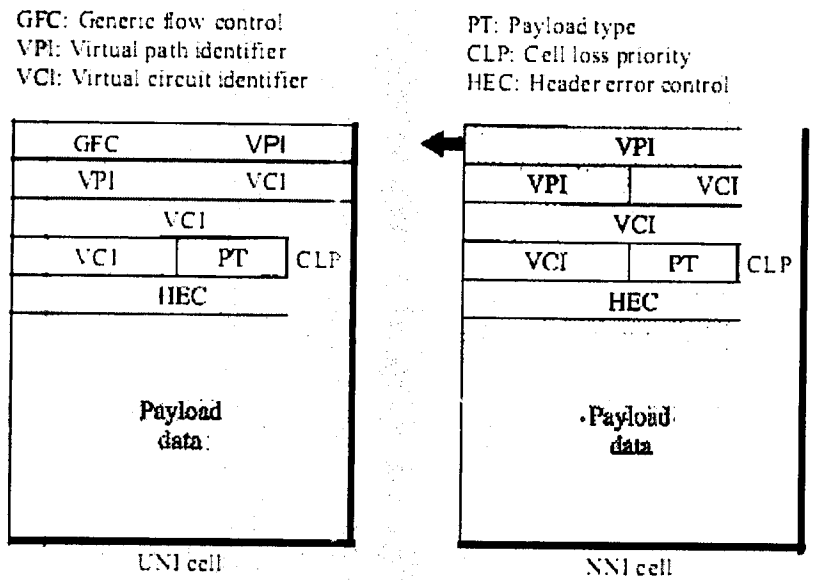


Figure 1.93 ATM headers

- Generic flow control (GFC). The 4-bit GFC field provides flow control at the UNI level. The ITU-T has determined that this level of flow control is not necessary at the NNI level. In the NNI header, therefore, these bits are added to the VPI. The longer VPI allows more virtual paths to be defined at the NNI level. The format for this additional VII has not yet been determined.
- Virtual path identifier (VPI). The VPI is an 8-bit field in a UNI cell and a 12-bit field in an NNI cell (see above).

- Virtual circuit identifier (VCI). The VCI is a 16-bit field in both frames.
- Payload type (PT). In the 3-bit PT field, the first bit defines the payload as user data or managerial information. The interpretation of the last 2 bits depends on the first bit.
- Cell loss priority (CLP). The 1-bit CLP field is provided for congestion control. A cell with its CLP bit set to I must be retained as long as there are cells with a CLP ofO. We discuss congestion control and quality of service in an ATM network in Chapter 24. Header error correction (HEC). The HEC is a code computed for the first 4 bytes of the header. It is a CRC with the divisor $x^8 + x^2 + x + 1$ that is used to correct single-bit errors and a large class of multiple-bit errors.

Application Adaptation Layer

The application adaptation layer (AAL) was designed to enable two ATM concepts. First, ATM must accept any type of payload, both data frames and streams ofbits. A data frame can come from an upper-layer protocol that creates a clearly defined frame to be sent to a carrier network such as ATM. A good exampre is the Internet. ATM must also carry multimedia payload. It can accept continuous bit streams and break them into chunks to be encapsulated into a cell at the ATM layer. AAL uses two iblayers to accomplish these tasks.

Whether the data are a data frame or a stream of bits, the payload must be segmented into 48-byte segments to be carried by a cell. At the destination, these segments need to be reassembled to recreate the original payload. The AAL defines a sublayer, called a segmentation and reassembly (SAR) sublayer, to do so. Segmentation is at the source; reassembly, at the destination.

Before data are segmented by SAR, they must be prepared to guarantee the integrity of the data. This is done by a sublayer called the convergence sub layer (CS).

ATM defines four versions of the AAL. AAL1, AAL2, AAL3/4, and AAL5. Although we discuss all these versions, we need to inform the reader that the common versions today are AAL 1 and AAL5. The first is used in streaming audio and video communication; the second, in data communications.

AAL1: AAL1 supports applications that transfer information at constant bit rates, such as video and voice. It allows ATM to connect existing digital telephone networks such as voice channels and T lines. Figure 1.94 shows how a bit stream of data is chopped into 47-byte chunks and encapsulated in cells.

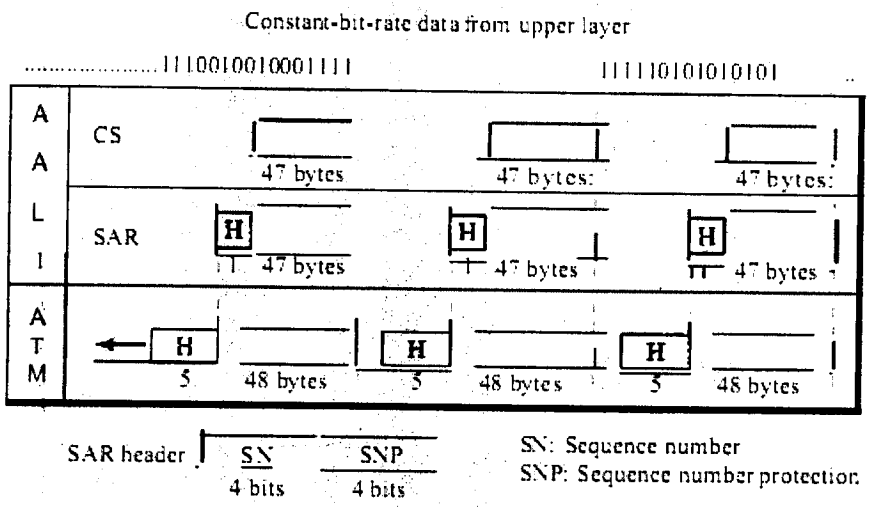


Figure 1.94 AAL1

The CS sub layer divides the bit stream into 47-byte segments and passes them to the SAR sublayer below. Note that the CS sublayer does not add a header.

The SAR sublayer adds 1 byte of header and passes the 48-byte segment to the ATM layer. The header has two fields:

- Sequence number (SN). This 4-bit field defines a sequence number to order the bits. The first bit is sometimes used for timing, which leaves 3 bits for sequencing (modulo 8).
- Sequence number protection (SNP). The second 4-bit field protects the first field. The first 3 bits automatically correct the SN field. The last bit is a parity bit that detects error over all 8 bits.

AAL2: Originally AAL2 was intended to support a variable-data-rate bit stream, but it has been redesigned. It is now used for low-bit-rate traffic and short-frame traffic such as audio (compressed or uncompressed), video, or tax. A good example of AAL2 use is in mobile telephony. AAL2 allows the multiplexing of short frames into one cell.

Figure 1.95 shows the process of encapsulating a short frame from the same source (the same user of a mobile phone) or from several sources (several users of mobile telephones) into one cell.

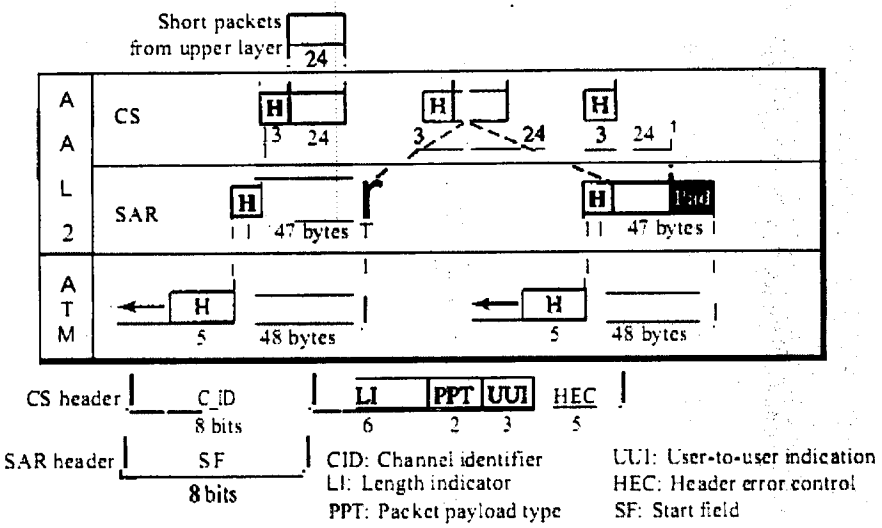


Figure 1.95 AAL2

The CS layer overhead consists of five fields:

- Channel identifier (CID): The 8-bit CID field defines the channel (user) of the short packet.
- Length indicator (LI): The 6-bit LI field indicates how much of the final packet is data.
- Packet payload type (PPT). The PPT field defines the type of packet.
- User-to-user indicator (UUI). The UUI field can be used by end-to-end users.
- Header error control (HE C). The last 5 bits is used to correct errors in the header.

The only overhead at the SAR layer is the start field (SF) that defines the offset from the beginning of the packet.

AAL3/4 Initially, AAL3 was intended to support connection-oriented data services and AAL4 to support connectionless services. As they evolved, however, it became evident that the fundamental issues of the two protocols were the same. They have therefore been combined into a single format calledAAL3/4. Figure 1.96 shows the AAL3/4 sublayer.

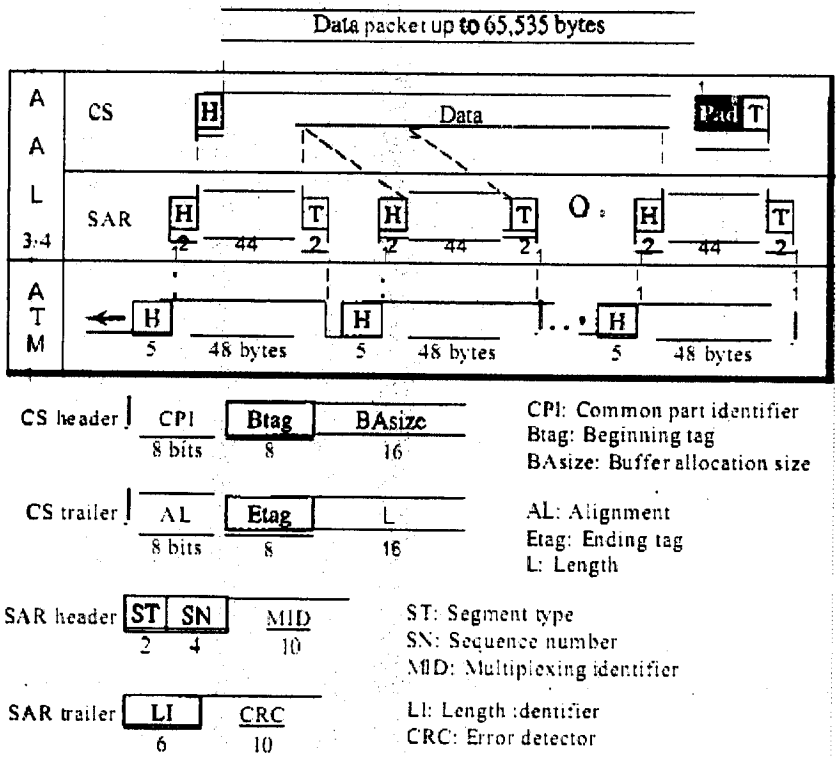


Figure 1.96 AAL 3/4

The CS layer header and trailer consist of six fields:

- Common part identifier (CPI). The CPI defines how the subsequent fields are to be interpreted. The value at present is 0.
- Begin tag (Stag). The value of this field is repeated in each CS to identify all the cells belonging to the same packet. The value is the same as the Etag (see below).
- Buffer allocation size (BAsize). The 2-byte BA field tells the receiver what size buffer is needed for the coming data.
- Alignment (AL). The 1-byte AL field is included to make the rest of the trailer 4 bytes long.
- Ending tag (Etag). The 1-byte ET field serves as an ending flag. Its value is the same as that of the beginning tag.
- Length (L). The 2-byte L field indicates the length of the data unit.

The SAR header and trailer consist of five fields:

- Segment type (ST). The 2-bit ST identifier specifies the position of the segment in the message: beginning (00), middle (01), or end (10). A single-segment message has an ST of 11.
- Sequence number (SN). This field is the same as defined previously.
- Multiplexing identifier (MID). The 10-bit MID field identifies cells coming from different data flows and multiplexed on the same virtual connection.
- Length indicator (LI). This field defines how much of the packet is data, not padding.
- CRC. The last 10 bits of the trailer is a CRC for the entire data unit.

AALS AAL3'4 provides comprehensive sequencing and error control mechanisms that are not necessary for every application. For these applications, the designers of ATM have provided a fifth AAL sublayer, called the simple and efficient adaptation layer (SEAL). AALS assumes that all cells belonging to a single message travel sequentially and that control functions are included in the upper layers of the sending application. Figure 1.97 shows the AAL5 sublayer.

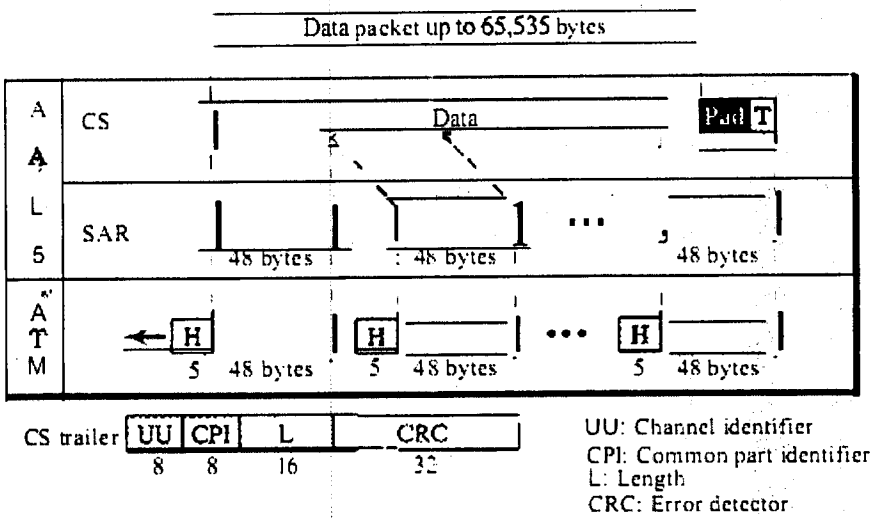


Figure 1.97 AAL5

The four trailer fields in the CS layer are

- User-to-user (UU). This field is used by end users, as described previously.
- Common part identifier (CPI). This field is the same as defined previously.
- Length (L). The 2-byte L field indicates the length of the original data.
- CRC. The last 4 bytes is for error control on the entire data unit.

1.14 Network Components

Almost all modern networks are created by connecting various physical devices, to establish a path from the sending to the receiving device. The most common groups: cables, hubs, bridges, switches, and routers.

Cables

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

Twisted-Pair Cable

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

One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two.

In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals.

If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e.g.. one is closer and the other is farther). This results in a difference at the receiver. By twist.ing the pairs, a balance is maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true.

Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals. The unwanted signals are mostly canceled out. From the above discussion, it is clear that the number of twists per unit of length (e.g., inch) has some effect on the quality of the cable.

Unshielded Versus Shielded Twisted-Pair Cable

The most common twisted-pair cable used in communications is referred to as unshielded twisted-pair (UTP). IBM has also produced a version of twisted-pair cable for its use called shielded twisted-pair (STP). STP cable has a metal foil or braided- mesh covering that encases each pair of insulated conductors. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive.

Categories

The Electronic Industries Association (EIA) has developed standards to classify unshielded twisted-pair cable into seven categories. Categories are determined by cable quality, with 1 as the lowest and 7 as the highest. Each EIA category is suitable for specific uses. Table 1.5 shows these categories.

Category	Specification	Data Rate (Mbps)	Use
1	Unshielded twisted-pair used in telephone	< 0.1	Telephone
2	Unshielded twisted-pair originally used in T-lines	2	T-lines
3	Improved CAT 2 used in LANs	10	LANs
4	Improved CAT 3 used in Token Ring networks	20	LANs
5	Cable wire is normally 24 AWG with a jacket and outside sheath	100	LANs
SE	An extension to category 5 that includes extra features to minimize the crosstalk and electromagnetic interference	125	LANs
6	A new category with matched components coming from the same manufacturer. The cable must be tested at a 200-Mbps data rate.	200	LANs
7	Sometimes called SSTP (shielded screen twisted-pair). Each pair is individually wrapped in a helical metallic foil followed by a metallic foil shield in addition to the outside sheath. The shield decreases the effect of crosstalk: and increases the data rate.	600	LANs

Table 1.5 Categories of unshielded twisted-pair cables

Hubs

Passive Hubs

A passive hub is just a connector. It connects the wires coining from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coining from different stations collide; the hub is the collision point. This type of a hub is part of the media; its location in the Internet model is below the physical layer.

Repeaters

A repeater is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern. The repeater then sends the refreshed signal. A repeater can extend the physical length of a LAN.

A repeater does not actually connect two LANs; it connects two segments of the same LAN. The segments connected are still part of one single LAN. A repeater is not a device that can connect two LANs of different protocols.

Active hubs

An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology.

Bridges

A bridge is a device that allows you to segment a large network into two smaller, more efficient networks. If you are adding to an older wiring scheme and want the new network to be up-to-date, a bridge can connect the two.

A bridge monitors the information traffic on both sides of the network so that it can pass packets of information to the correct location. Most bridges can listen" to the network and automatically figure out the address of each computer on both sides of the bridge. The bridge can inspect each message and, if necessary, broadcast it on the other side of the network.

The bridge manages the traffic to maintain optimum performance on both sides of the network. You might say that the bridge is like a traffic cop at a busy intersection during rush hour. It keeps information flowing on both sides of the network, but it does not allow unnecessary traffic through. Bridges can be used to connect different types of cabling, or physical topologies. They must, however, be used between networks with the same protocol.

Switch

A Switch is more sophisticated than hub and can remember and check node addresses. In fact this phenomenon can affect logical topology of the network! They physically resemble hubs and like hubs, they vary in number of ports, stand-alone vs. stackable, and managed vs. unmanaged.

While a hub broadcasts data frames to all ports, the switch reads the destination address of the data frame and only sends it to the corresponding port. The effect is to turn the network into a group of point-to-point circuits and thus changes the logical topology of the network from a bus to a star.

When a switch is first turned on, its internal forwarding table is empty. It then learns which ports correspond to which computers by reading the source addresses of the incoming frames along with the port number that the frame arrived on. If the switch's forwarding table does not have the destination address of the data frame, it broadcasts the frame to all ports.

Thus, a switch starts by working like a hub and then works more and more as a switch as it fills its forwarding table. Thus they work at machine address level.

Router:

A router translates information from one network to another; it is similar to a super intelligent bridge. Routers select the best path to route a message, based on the destination address and origin. The router can direct traffic to prevent head-on collisions, and is smart enough to know when to direct traffic along back roads and shortcuts.

While bridges know the addresses of all computers on each side of the network, routers know the addresses of computers, bridges, and other routers on the network. Routers can even "listen" to the entire network to determine which sections are busiest they can then redirect data around those sections until they clear up.

If you have a school LAN that you want to connect to the Internet, you will need to purchase a router. In this case, the router serves as the translator between the information on your LAN and the Internet. It also determines the best route to send the data over the Internet.

Routers can:

- Direct signal traffic efficiently
- Route messages between any two protocols
- Route messages between linear bus, star, and star wired ring topologies
- Route messages across fiber optic, coaxial, and twisted-pair cabling

Summary

In this unit, you have learnt about data communication. Data communications are the transfer of data from one device to another via some form of transmission medium. A data communications system must transmit data to the correct destination in an accurate and timely manner. The five components that make up a data communications system are the message, sender, receiver, medium, and protocol. Text, numbers, images, audio, and video are different forms of information.

Data communications concerns itself with the transmission (sending and receiving) of information between two locations by means of electrical signals. The media can be guided (physical) wires or unguided (radio links). The two types of electrical signals are analog and digital.

Analogue signals have three main characteristics which define them, being amplitude, frequency and phase. Speech is an example of an analogue signal.

In data communications the channel bandwidth is represented in bits/sec and Shannon's theorem imposes a bandwidth restriction for low noise transmission.

Many aspects of data transmission are covered in this unit. There are six types of communication point-point, point-multipoint, broadcast, simplex, half duplex and full duplex.

Two modes of transmission are synchronous and asynchronous. Protocols (mutually agreed set of rules) are necessary to implement this type of serial communication system.

The network topology defines how the devices (computers, printers...etc) are connected and how the data flows from one device to another. They are broadly categorized as bus, ring, star, mesh and hybrid. In a bus topology all devices are connected to the transmission medium as backbone. Ring topology was in the beginning of LAN area. In a star topology each station is connected to a central node. The central node can be either a hub or a switch. A mesh physical topology is when every device on the network is connected to every device on the network; most commonly used in WAN configurations. A hybrid topology is a combination of any two or more network topologies in such a way that the resulting network does not have one of the standard forms.

Review Questions

1. Explain the components of data communication.
2. What are the characteristics of data communication?
3. Differentiate between an analog and a digital signal. .
4. Define analog and digital signals
5. Explain Composite analog signals.
6. Explain Time and Frequency Domain Representation of signals
7. Explain the characteristics of an Analog signal
8. Explain the characteristics of a Digital signal
9. Explain flow control in data communication.
10. Explain the different modes of data transmission.
11. What is modulation? What are the various modulation techniques?
12. What are the different types of error detecting and correcting codes?
13. Define computer network and categorize.
14. What is the OSI model? List its layers and explain their responsibility in exactly one line.
15. Explain how the communication takes place between layers of OSI model.
16. Differentiate between the working of Data link layer, Network layer and Transport layer.
17. What is multiplexing? Explain the different types of multiplexing,
18. Explain the various network topologies.
19. Why you need computer networking? What are the disadvantages OF computer networking?

Reference Books

1. Computer Networking: Schaum's outlines (TMH).
2. Kurose J F & Ross K.W: Computer Networking (Pearson)
3. Tanenbaum A S: Computer Networks (PHI) 4th Ed.
4. Data Communication & Networking – Behrouz Forouzan(TMH)

UNIT – II

Structure

2.1 Introduction

In the last unit, you learnt about data communication and computer networks and in this unit, you will learn about network technologies. Network technology, includes types of networking such as Local Area Network (LAN), Metropolitan Area Network (MAN), Wide Area Network (WAN) and the IEEE standards for LAN and WAN. In addition to this, you will also study the various transmission media used in communication channels. The two types of media of communication channels that help transfer data are:

- Wired transmission medium
- Wireless transmission medium

Wired transmission medium is also called guided transmission medium as it performs communication with the help of a physical medium. The various wired transmission media include twisted pair cable, coaxial cable and

fiber-optic cable. Wireless media do not require a\ physical medium for data communication they transmit data with the help of satellites, using antennae for radiating as we as receiving signals. The various types of wireless transmission include microwave, laser and radio. This unit will also introduce you to the multiple-access techniques further temporary access of network to the user such as Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA) and various protocols used for channel allocation such as ALOHA and CSMA, There are two version of ALOHA:

• Pure ALOHA

• Slotted ALOHA

Carrier Sense Multiple Access/ Collision Detection (CSMA/CD) is a set of rule that is used to determine the response of the network stations, when they access the same data channel simultaneously.

2.2 Multiple Access Channels (MAC)

We are all familiar with the notion of broadcasting, as television has been using it since its invention. But traditional television is a one-way broadcast (i.e., one fixed node transmitting to many receiving nodes), while nodes on a computer network broadcast channel can both send and receive. Perhaps a more apt human analogy for a broadcast channel is a cocktail party, where many people gather together in a large room (the air providing the broadcast medium) to talk and listen. A second good analogy is something many readers will be familiar with - a classroom where teacher(s) and student(s) similarly share the same, single, broadcast medium. A central problem in both scenarios is that of determining who gets to talk (i.e., transmit into the channel), and when. As humans, we've evolved an elaborate set of protocols for sharing the broadcast channel ("Give everyone a chance to speak." "Don't speak until you are spoken to." "Don't monopolize the conversation." "Raise your hand if you have question." "Don't interrupt when someone is speaking." "Don't fall asleep when someone else is talking.").

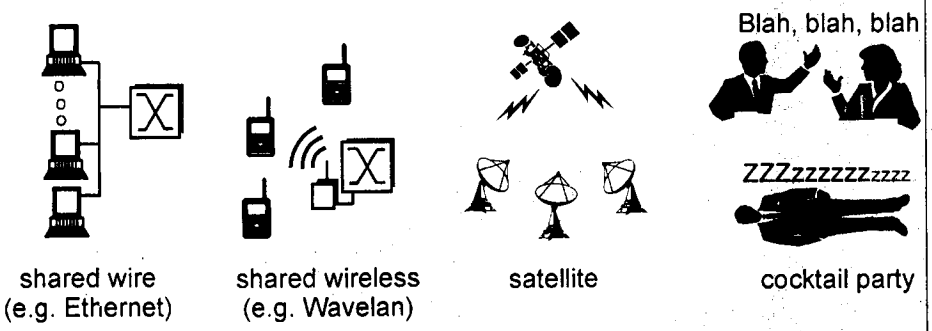


Figure 2.1: Various multiple access channels

Computer networks similarly have protocols - so-called multiple access protocols by which nodes regulate their transmission onto the shared broadcast channel. As shown in Figure 2.1, multiple access protocols are needed in a wide variety of network settings, including both wired and wireless local area networks, and satellite networks. Figure 2.2 takes a more abstract view of the broadcast channel and of the nodes sharing that channel. Although technically each node accesses the broadcast channel through its adapter, in this section we will refer to the node as the sending and receiving device. In practice, hundreds or even thousands of nodes can directly communicate over a broadcast channel.

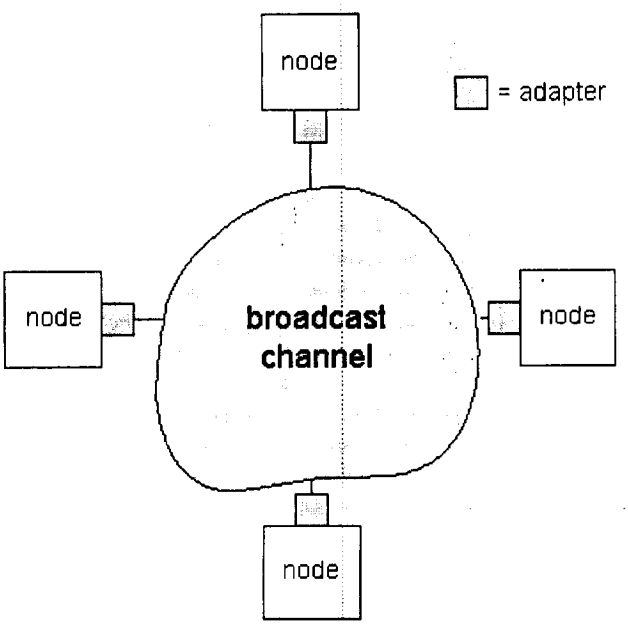


Figure 2.2: A broadcast channel interconnecting four nodes.

Because all nodes are capable of transmitting frames, more than two nodes can transmit frames at the same time. When this happens, all of the nodes receive multiple frames at the same time, that is, the transmitted frames collide at all of the receivers. Typically, when there is a collision, none of the receiving nodes can make any sense of any of the frames that were transmitted; in a sense, the signals of the colliding frame become inextricably tangled together. Thus, all the frames involved in the collision are lost, and the broadcast channel is wasted during the collision interval. Clearly, if many nodes want to frequently transmit frames, many transmissions will result in collisions, and much of the bandwidth of the broadcast channel will be wasted.

In order to ensure that the broadcast channel performs useful work when multiple nodes are active, it is necessary to somehow coordinate the transmissions of the active nodes. This coordination job is the responsibility of the multiple access protocol. Over the past thirty years, thousands of papers and hundreds of Ph.D. dissertations have been written on multiple access protocols; a comprehensive survey of this body of work is Furthermore, dozens of different protocols have been implemented in a variety of link-layer technologies. Nevertheless, we can classify just about any multiple access protocol as belonging to one of three categories: channel partitioning protocols, random access protocols, and taking-turns protocols. We'll cover these categories of multiple access protocols in the following three subsections. Let us conclude this overview by noting that ideally, a multiple access protocol for a broadcast channel of rate R bits per second should have the following desirable characteristics:

When only one node has data to send, that node has a throughput of R bps.

When M nodes have data to send, each of these nodes has a throughput of R/M bps. This need not necessarily imply that each of the M nodes always have an instantaneous rate of R/M , but rather that each node should have an average transmission rate of R/M over some suitably-defined interval of time.

The protocol is decentralized, i.e., there are no master nodes that can fail and bring down the entire system.

The protocol is simple, so that it is inexpensive to implement.

2.3 Channel Partitioning Protocols

Recall from our early discussion back in unit-I, that Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM) are two techniques that can be used to partition a broadcast channel's bandwidth among all nodes sharing that channel. As an example, suppose the channel supports N nodes and that the transmission rate of the channel is R bps. TDM divides time into time frames (not to be confused the unit of data, the frame, at the data link layer) and further divides each time frame into N time slots. Each slot time is then assigned to one of the N nodes. Whenever a node has a frame to send, it transmits the frame's bits during its assigned time slot in the revolving TDM frame. Typically, frame sizes are chosen so that a single frame can be transmitting during a slot time. Figure 2.3 shows a simple four-node TDM example. Returning to our cocktail party analogy, a TDM-regulated cocktail party would allow one partygoer to speak for a fixed period of time, and then allow another partygoer to speak for the same amount of time, and so on. Once everyone has had their chance to talk, the pattern repeats.

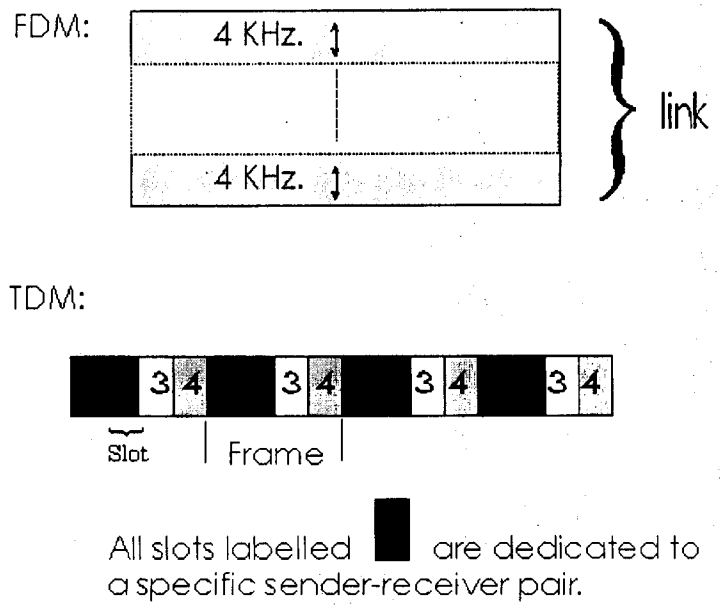


Figure 2.3: A four-node TDM and FDM example

TDM is appealing as it eliminates collisions and is perfectly fair: each node gets a dedicated transmission rate of R/N bps during each slot time. However, it has two major drawbacks. First, a node is limited to this rate of R/N bps over a slot's time even when it is the only node with frames to

send. A second drawback is that a node must always wait for its turn in the transmission sequence again, even when it is the only node with a frame to send. Imagine the partygoer who is the only one with anything to say (and imagine that this is the even rarer circumstance where everyone at the party wants to hear what that one person has to say). Clearly, TDM would be a poor choice for a multiple access protocol for this particular party.

While TDM shares the broadcast channel in time, FDM divides the R bps channel into different frequencies (each with a bandwidth of R/N) and assigns each frequency to one of the N nodes. FDM thus creates N "smaller" channels of R/N bps out of the single, "larger" R bps channel. FDM shares both the advantages and drawbacks of TDM. It avoids collisions and divides the bandwidth fairly among the N nodes. However, FDM also shares a principal disadvantage with TDM - a node is limited to a bandwidth of R/N, even when it is the only node with frames to send.

A third channel partitioning protocol is Code Division Multiple Access (CDMA). While TDM and FDM assign times slots and frequencies, respectively, to the nodes, CDMA assigns a different code to each node. Each node then uses its unique code to encode the data bits it sends, as discussed below. We'll see that CDMA allows different nodes to transmit simultaneously and yet have their respective receivers correctly receive a sender's encoded data bits (assuming the receiver knows the sender's code) in spite of "interfering" transmissions by other nodes. CDMA has been used in military systems for some time (due its anti jamming properties) and is now beginning to find widespread civilian use, particularly for uses in wireless multiple access channels.

In a CDMA protocol, each bit being sent by the sender is encoded by multiplying the bit by a signal (the code) that changes at a much faster rate (known as the chipping rate) than the original sequence of data bits. Figure 2.4 shows a simple, idealized CDMA encoding/decoding scenario. Suppose that the rate at which original data bits reach the CDMA encoder defines the unit of time; that is, each original data bit to be transmitted requires one bit-slot time. Let d_i be the value of the data bit for the ith bit slot. Each bit slot is further subdivided into M mini-slots; in Figure 5.2.4, M=8, although in practice M is much larger. The CDMA code used by the sender consists of a sequence of M values, c_m, m = 1... M, each taking a +1 or -1 value. In the example in Figure 5.2.4, the M-bit CDMA code being used by the sender is (1, 1, 1, -1, 1, -1, -1, -1).

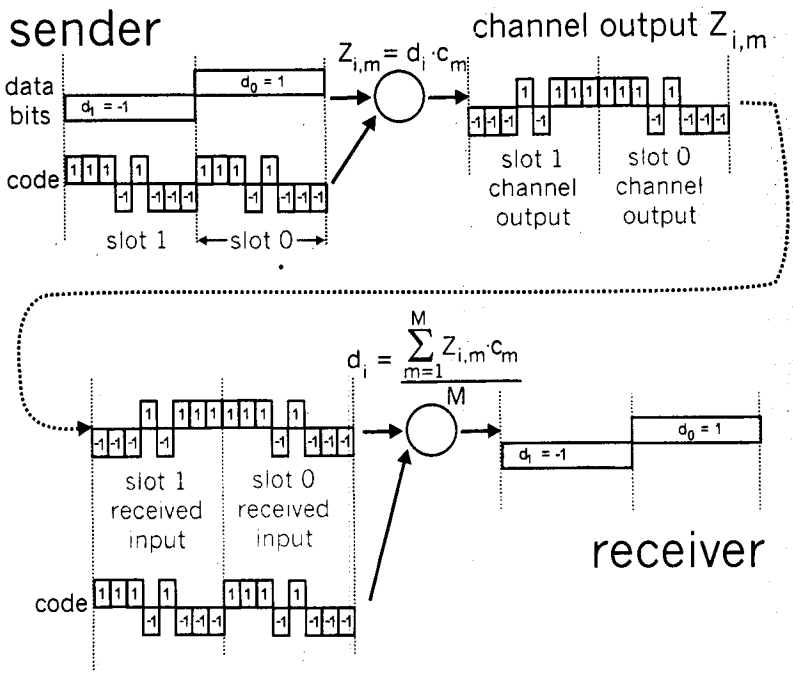


Figure 2.4: A simple CDMA example: sender encoding, receiver decoding

To illustrate how CDMA works, let us focus on the ith data bit, d_i. For the mth mini-slot of the bit-transmission time of d_i, the output of the CDMA encoder, $Z_{i,m}$, is the value of d_i multiplied by the mth bit in the assigned CDMA code, c_m:

$Z_{i,m} = d\, c_m$ (Equation 2.1)

In a simple world, with no interfering senders, the receiver would receive the encoded bits, $Z_{i,m}$, and recover the original data bit, d_i, by computing:

$d_i = (1/M) \sum_{m=1}^{M} Z_{i,m}\ c_m$ (Equation 2.2)

The reader might want to work through the details of the example in Figure 2.4 to see that the original data bits are indeed correctly recovered at the receiver using Equation 2.2

The world is far from ideal, however, and as noted above, CDMA must work in the presence of interfering senders that are encoding and transmitting their data using a different assigned code. But how can a CDMA receiver recover a sender's original data bits when those data bits are being tangled with bits being transmitted by other senders? CDMA works under the assumption that the interfering transmitted bit signals are additive, e.g., that if three senders send a 1 value, and a fourth sender

sends a -1 value during the same mini-slot, then the received signal at all receivers during hat mini-slot is a 2 (since 1+ 1 + 1 - 1 = 2). In the presence of multiple senders, sender s computes its encoded transmissions, $Z_{i,m}{}^{s}$, in exactly the same manner as in Equation 5.2.1. The value received at a receiver during the mth minis lot of the ith bit slot, however, is now the sum of the transmitted bits from all N senders during that minis lot:

$$Z_{i,m}{}^{*} = \sum_{s=1}^{N} Z^{s}_{.m}$$

Amazingly, if the senders' codes are chosen carefully, each receiver can recover the data sent by a given sender out of the aggregate signal simply by using the sender's code in exactly the same manner as in Equation 2.2:

$$d_i = (1/M) \sum_{m=1}^{M} Z_{im}{}^{*} \cdot c_m \qquad \text{(Equation 2.3)}$$

Figure 2.5 illustrates a two-sender CDMA example. The M-bit CDMA code being used by the upper sender is (1, 1, 1, -1, 1, -1, -1, -1), while the CDMA code being used by the lower sender is (1, -1, 1, 1, 1, -1, 1, 1). Figure 2.5 illustrates a receiver recovering the original data bits from the upper sender. Note that the receiver is able to extract the data from sender 1 in spite of the interfering transmission from sender 2. Returning to our cocktail party analogy, a CDMA protocol is similar to having partygoers speaking in multiple languages; in such circumstances humans are actually quite good at locking into the conversation in the language they understand, while filtering out the remaining conversations.

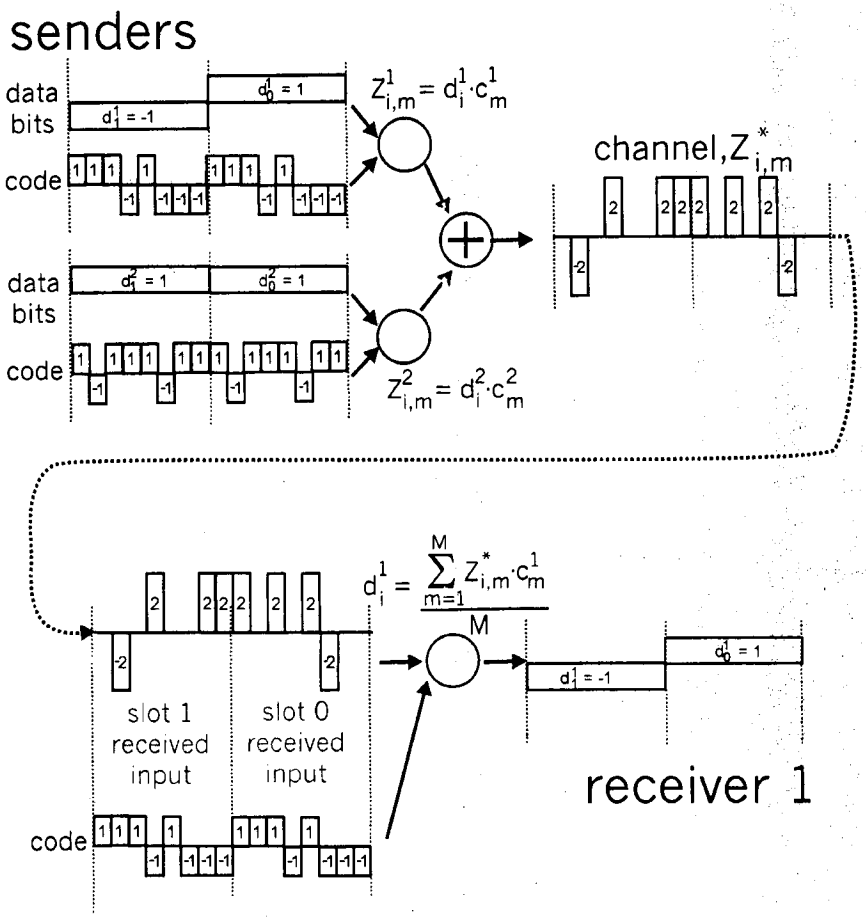


Figure 2.5: A two-sender CDMA example

Our discussion here of CDMA is necessarily brief and a number of difficult issues must be addressed in practice. First, in order for the CDMA receivers to be able to extract out a particular sender's signal, the CDMA codes must be carefully chosen. Secondly, our discussion has assumed that the received signal strengths from various senders at a receiver are the same; this can be difficult to achieve in practice. There is a considerable body of literature addressing these and other issues related to CDMA.

2.4 Random Access Protocols

The second broad class of multiple access protocols is so-called random access protocols. In a random access protocol, a transmitting node always transmits at the full rate of the channel, namely, R bps. When there is a collision, each node involved in the collision repeatedly retransmits its frame until the frame gets through without a collision. But when a node

experiences a collision, it doesn't necessarily retransmit the frame right away. Instead it waits a random delay before retransmitting the frame. Each node involved in a collision chooses independent random delays. Because after a collision the random delays are independently chosen, it is possible that one of the nodes will pick a delay that is sufficiently less than the delays of the other colliding nodes, and will therefore be able to "sneak" its frame into the channel without a collision.

Slotted ALOHA

Let's begin our study of random access protocols with one of the most simple random access protocols, the so-called slotted ALOHA protocol. In our description of slotted ALOHA, we assume the following:

All frames consist of exactly L bits.

Time is divided into slots of size L/R seconds (i.e., a slot equals the time to transmit one frame).

Nodes start to transmit frames only at the beginnings of slots.

The nodes are synchronized so that each node knows when the slots begin.

If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends.

Let p be a probability, that is, a number between 0 and 1. The operation of slotted ALOHA in each node is simple:

When the node has a fresh frame to send, it waits until the beginning of the next slot and transmits the entire frame in the slot.

If there isn't a collision, the node won't consider retransmitting the frame. (The node can prepare a new frame for transmission, if it has one.)

If there is a collision, the node detects the collision before the end of the slot. The node retransmits its frame in each subsequent slot with probability p until the frame is transmitted without a collision.

By retransmitting with probability p, we mean that the node effectively tosses a biased coin; the event heads corresponds to retransmit, which occurs with probability p. The event tails corresponds to "skip the slot and

toss the coin again in the next slot"; this occurs with probability (1-p). Each of the nodes involved in the collision toss their coins independently.

Slotted ALOHA would appear to have many advantages. Unlike channel partitioning, slotted ALOHA allows a single active node (i.e., a node with a frame to send) to continuously transmit frames at the full rate of the channel. Slotted ALOHA is also highly decentralized, as each node detects collisions and independently decides when to retransmit. (Slotted ALOHA does, however, require the slots to be synchronized in the nodes; we'll shortly discuss an unslotted version of the ALOHA protocol, as well as CSMA protocols; one of which require such synchronization and are therefore fully decentralized.) Slotted ALOHA is also an extremely simple protocol.

Slotted ALOHA also works great when there is only one active node, but how efficient is it when there are multiple active nodes? There are two possible efficiency concerns here. First, as shown in Figure 2.6, when there are multiple active nodes, a certain fraction of the slots will have collisions and will therefore be "wasted." The second concern is that another fraction of the slots will be empty because all active nodes refrain from transmitting as a result of the probabilistic transmission policy. The only "un-wasted" slots will be those in which exactly one node transmits. A slot in which exactly one node transmits is said to be a successful slot. The efficiency of a slotted multiple access protocol is defined to be the long-run fraction of successful slots when there are a large number of active nodes, with each node having a large number of frames to send. Note that if no form of access control were used, and each node were to immediately retransmits after each collision, the efficiency would be zero. Slotted ALOHA clearly increases the efficiency beyond zero, but by how much?

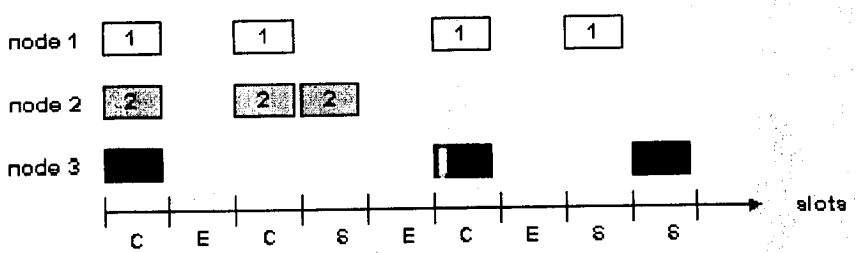


Figure 2.6: Nodes 1, 2 and 3 collide in the first slot. Node 2 finally succeeds in the fourth slot, node 1 in the eighth slot, and node 3 in the ninth slot. The notation C, E and S represent "collision slot", "empty slot" and "successful slot", respectively

We now proceed to outline the derivation of the maximum efficiency of slotted ALOHA. To keep this derivation simple, let's modify the protocol a little and assume that each node attempts to transmit a frame in each slot with probability p. (That is, we assume that each node always has a frame to send and that the node transmits with probability p for a fresh frame as well as for a frame that has already suffered a collision.) Suppose first there are N nodes. Then the probability that a given slot is a successful slot is the probability that one of the nodes transmits and that the remaining N-1 nodes do not transmit. The probability that a given node transmits is p; the probability that the remaining nodes do not transmit is $(1-p)^{N-1}$. Therefore the probability a given node has a success is $p\ (1-p)^{N-1}$. Because there are N nodes, the probability that an arbitrary node has a success is $Np(1-p)^{N-1}$.

Thus, when there are N active nodes, the efficiency of slotted ALOHA is $Np(1-p)^{N-1}$. To obtain the maximum efficiency for N active nodes, we have to find the p^* that maximizes this expression. (See the homework problems for a general outline of this derivation.) And to obtain the maximum efficiency for a large number of active nodes, we take the limit of $Np^*(1-p^*)^{N-1}$ as N approaches infinity. (Again, see homework problems.) After performing these calculations, we'll find that the maximum efficiency of the protocol is given by 1/e = .37. That is, when a large number of nodes have many frames to transmit, then (at best) only 37% of the slots do useful work. Thus the effective transmission rate of the channel is not R bps but only .37 R bps! A similar analysis also shows that 37% of the slots go empty and 26% of slots have collisions. Imagine the poor network administrator who has purchased a 100 Mbps slotted ALOHA system, expecting to be able to use the network to transmit data among a large number of users at an aggregate rate of, say, 80 Mbps! Although the channel is capable of transmitting a given frame at the full channel rate of 100Mbps, in the long term, the successful throughput of this channel will be less that 37 Mbps.

ALOHA

The slotted ALOHA protocol required that all nodes synchronize their transmissions to start at the beginning of a slot. The first ALOHA protocol [Abramson 1970] was actually an unslotted, fully decentralized, protocol. In so-called pure ALOHA, when a frame first arrives (i.e., a network layer datagram is passed down from the network layer at the sending node), the node immediately transmits the frame in it's entirely into the broadcast channel. If a transmitted frame experiences a collision with one or more

other transmissions, the node will then immediately (after completely transmitting its collided frame) retransmit the frame with probability p. Otherwise, the node waits for a frame transmission time. After this wait, it then transmits the frame with probability p, or waits (remaining idle) for another frame time with probability 1-p.

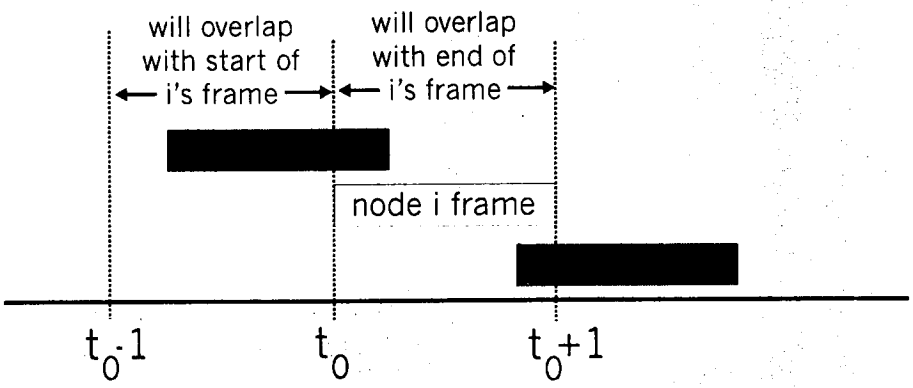


Figure 2.7: Interfering transmissions in pure Aloha

To determine the maximum efficiency of pure ALOHA, we focus on an individual node. We'll make the same assumptions as in our slotted ALOHA analysis and take the frame transmission time to be the unit of time at any given time, the probability that a node is transmitting a frame is p. suppose this frame begins transmission at time t_0. As shown in Figure 2.7, in order for this frame to be successfully transmitted, no other nodes can begin their transmission in the interval of time $[t_0-1, t_0]$. Such a transmission would overlap with the beginning of the transmission of node i's frame. The probability that all other nodes do not begin a transmission in this interval is $(1-p)^{N-1}$. Similarly, no other node can begin a transmission while node i is transmitting, as such a transmission would overlap with the latter part of node i's transmission. The probability that all other nodes do not begin a transmission in this interval is also $(1-p)^{N-1}$. Thus, the probability that a given node has a successful transmission is $p(1-p)^{2(N-1)}$. By taking limits as in the slotted ALOHA case, we find that the maximum efficiency of the pure ALOHA protocol is only 1/(2e) - exactly half that of slotted ALOHA. This then is the price to be paid for a fully decentralized ALOHA protocol.

CSMA - Carrier Sense Multiple Access

In both slotted and pure ALOHA, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel. In particular, a node neither pays attention to whether another

node happens to be transmitting when it begins to transmit, nor stops transmitting if another node begins to interfere with its transmission. In our cocktail party analogy, ALOHA protocols are quite like a boorish partygoer who continues to chatter away regardless of whether other people are talking. As humans, we have human protocols that allow allows us to not only behave with more civility, but also to decrease the amount of time spent "colliding" with each other in conversation and consequently increasing the amount of amount of data we exchange in our conversations. Specifically, there are two important rules for polite human conversation:

Listen before speaking. If someone else is speaking, wait until they are done. In the networking world, this is termed carrier sensing - a node listens to the channel before transmitting. If a frame from another node is currently being transmitted into the channel, a node then waits ("backs off") a random amount of time and then again senses the channel. If the channel is sensed to be idle, the node then begins frame transmission. Otherwise, the node waits another random amount of time and repeats this process.

If someone else begins talking at the same time, stop talking. In the networking world, this is termed collision detection - a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering frame, it stops transmitting and uses some protocol to determine when it should next attempt to transmit.

These two rules are embodied in the family of CSMA (Carrier Sense Multiple Access) and CSMA/CD (CSMA with Collision Detection) protocols. Many variations on CSMA and CSMA/CD have been proposed, with the differences being primarily in the manner in which nodes perform backoff. The reader can consult these references for the details of these protocols.

The first question that one might ask about CSMA is that if all nodes perform carrier sensing, why do collisions occur in the first place? After all, a node will refrain from transmitting whenever it senses that another node is transmitting. The answer to the question can best be illustrated using space-time diagrams. Figure 2.7 shows a space-time diagram of four nodes (A, B, C, D) attached to a linear broadcast bus. The horizontal axis shows the position of each node in space; the y-axis represents time.

At time t_0, node B senses the channel is idle, as no other nodes are currently transmitting. Node B thus begins transmitting, with its bits propagating in both directions along the broadcast medium. The downward propagation of B's bits in Figure 2.7 with increasing time indicates that a non-zero amount of time is needed for B's bits to actually propagate (albeit at near the speed-of-light) along the broadcast medium. At time t_1 ($t_1 > t_0$), node D has a frame to send. Although node B is currently transmitting at time t_1, the bits being transmitted by B have yet to reach D, and thus D senses the channel idle at t_1. In accordance with the CSMA protocol, D thus begins transmitting its frame. A short time later, B's transmission begins to interfere with D's transmission at D. From Figure 2.7, it is evident that the end-to-end channel propagation delay of a broadcast channel - the time it takes for a signal to propagate from one of the channel to another will play a crucial role in determining its performance. Longer this propagation delay, the larger the chance that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network.

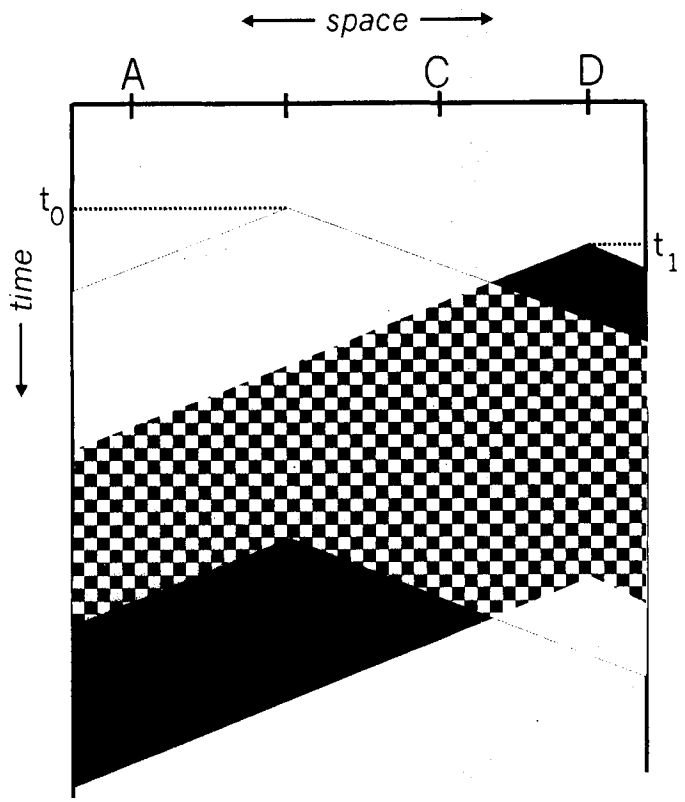


Figure 2.7: Space-time diagram of two CSMA nodes with colliding transmissions

In Figure 2.7, nodes do not perform collision detection; both B and D continue to transmit their frames in their entirety even though a collision has occurred. When a node performs collision detection it will cease transmission as soon as it detects a collision. Figure 2.8 shows the same scenario as in Figure 2.7, except that the two nodes each abort their transmission a short time after detecting a collision. Clearly, adding collision detection to a multiple access protocol will help protocol performance by not transmitting a useless, damaged (by interference with a frame from another node) frame in its entirety. The Ethernet protocol we will study in lesson-4 is a CSMA protocol that uses collision detection.

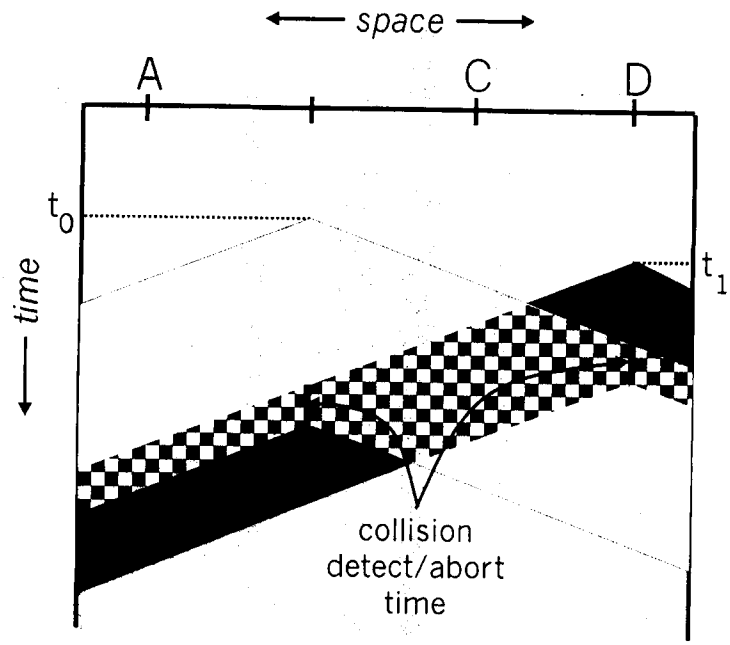


Figure 2.8: CSMA with collision detection.

2.5 Taking-Turns Protocols

Recall that two desirable properties of a multiple access protocol are (i) when only one node is active, the active node has a throughput of R bps, and (ii) when M nodes are active, then each active node has a throughput of nearly R/M bps. The ALOHA and CSMA protocols have this first property but not the second. This has motivated researchers to create another class of protocols the taking-turns protocols. As with random-access protocols, there are dozens of taking-turns protocols and each one of these protocols has many variations. We'll discuss two of the more important protocols here. The first one is the polling protocol. The polling protocol requires one of the nodes to be designated as a "master node" (or requires the

introduction of a new node serving as the master). The master node polls each of the nodes in a round-robin fashion. In particular, the master node first sends a message to node 1, saying that it can transmit up to some maximum number of frames. After node 1 transmits some frames (from zero up to the maximum number), the master node tells node 2 it can transmit up to the maximum number of frames. (The master node can determine when a node has finished sending its frames by observing the lack of a signal on the channel.) The procedure continues in this manner, with the master node polling each of the nodes in a cyclic manner.

The polling protocol eliminates the collisions and the empty slots that plague the random access protocols. This allows it to have a much higher efficiency. But it also has a few drawbacks. The first drawback is that the protocol introduces a polling delay, the amount of time required to notify a node that it can transmit. If, for example, only one node is active, then the node will transmit at a rate less than R bps, as the master node must poll each of the inactive nodes in turn, each time the active node sends its maximum number of frames. The second drawback, which is potentially more serious, is that if the master node fails, the entire channel becomes inoperative.

The second taking-turn protocol is the token-passing protocol. In this protocol there is no master node. A small, special-purpose frame known as a token is exchanged among the nodes in some fixed order. For example, node 1 might always send the token to node 2, node 2 might always send the token to node 3, and node N might always send the token to node 1. When a node receives a token, it holds onto the token only if it has some frames to transmit; otherwise, it immediately forwards the token to the next node. If a node does have frames to transmit when it receives the token, it sends up to a maximum number of frames and then forwards the token to the next node. Token passing is decentralized and has a high efficiency. But it has its problems as well. For example, the failure of one node can crash the entire channel. Or if a node accidentally neglects to release the token, then some recovery procedure must be invoked to get the token back in circulation? Over the years many token-passing products have been developed, and each one had to address these as well as other sticky issues.

2.6 Local Area Networks

Multiple access protocols are used in conjunction with many different types of broadcast channels. They have been used for satellite and wireless channels, whose nodes transmit over a common frequency spectrum. They are currently used in the upstream channel for cable access to the Internet. And they are extensively used in local area networks (LANs).

Recall that a LAN is a computer network that is concentrated in a geographical area, such as in a building or on a university campus. When a user accesses the Internet from a university or corporate campus, the access is almost always by way of a LAN. For this type of Internet access, the user's host is a node on the LAN, and the LAN provides access to the Internet through a router, as shown in Figure 2.9. The LAN is a single "link" between each user host and the router; it therefore uses a link-layer protocol, which incorporates a multiple access protocol. The transmission rate, R, of most LANs is very high. Even in the early 1980s, 10 Mbps LANs were common; today, 100 Mbps LANs are common, and 1 Gbps LANs are available.

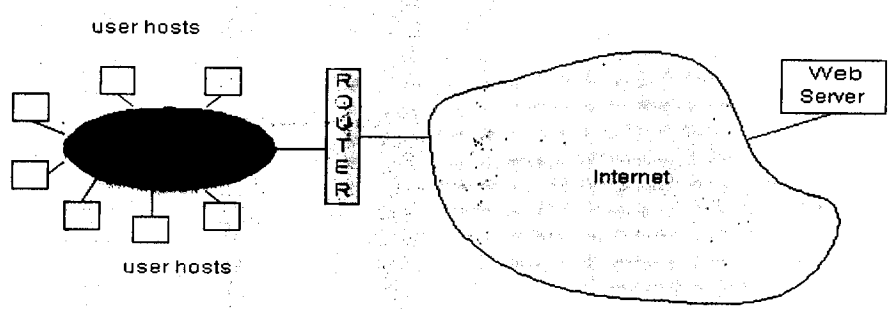


Figure 2.9: User hosts access an Internet Web server through a LAN. The broadcast channel between a user host and the router consists of one "link".

In the 1980s and the early 1990s, two classes of LAN technologies were popular in the workplace. The first class consists of the Ethernet LANs (also known as 802.3 LANs, which are random-access based. The second class of LAN technologies is token-passing technologies, including token ring (also known as IEEE 802.5 and FDDI (also known as Fiber Distributed Data Interface). Because we shall explore the Ethernet technologies in some detail in lesson-4, we focus our discussion here on the token-passing LANs. Our discussion on token-passing technologies is intentionally brief, since these technologies have become relatively minor players in the face of relentless Ethernet competition. Nevertheless, in order to provide

examples about token-passing technology and to give a little historical perspective, it is useful to say a few words about token rings.

In a token ring LAN, the N nodes of the LAN (hosts and routers) are connected in a ring by direct links. The topology of the token ring defines the token-passing order. When a node obtains the token and sends a frame, the frame propagates around the entire ring, thereby creating a virtual broadcast channel. The node that sends the frame has the responsibility of removing the frame from the ring. FDDI was designed for geographically larger LANs (so called MANs, that is, metropolitan area networks). For geographically large LANs (spread out over several kilometers) it is inefficient to let a frame propagate back to the sending node once the frame has passed the destination node. FDDI has the destination node remove the frame from the ring. (Strictly speaking, FDDI is not a pure broadcast channel, as every node does not receive every transmitted frame.)

Ethernet has pretty much taken over the LAN market. As recently as the 1980s and the early 1990s, Ethernet faced many challenges from other LAN technologies, including token ring, FDDI and ATM. Some of these other technologies succeeded at capturing a part of the market share for a few years. But since its invention in the mid-1970, Ethernet has continued to evolve and grow, and has held on to its dominant market share. Today, Ethernet is by far the most prevalent LAN technology, and is likely to remain so for the foreseeable future. One might say that Ethernet has been to local area networking what the Internet has been to global networking:

There are many reasons for Ethernet's success. First, Ethernet was the first widely-deployed high-speed LAN. Because it was deployed early, network administrators became intimately familiar with Ethernet its wonders and its quirks and were reluctant to switch over to other LAN technologies when they came on the scene. Second, token ring, FDDI and ATM are more complex and expensive than Ethernet, which further discouraged network administrators from switching over. Third, the most compelling reason to switch to another LAN technology (such as FDDI or ATM) was usually the higher data rate of the new technology; however, Ethernet always fought back, producing versions that operated at equal data rates or higher. Switched Ethernet was also introduced in the early 1990s, which further increased its effective data rates. Finally, because Ethernet has been so popular, Ethernet hardware (in particular, network interface cards) has become a commodity and is remarkably cheap. This low cost is also due o

the fact that Ethernet's multiple access protocol, CSMA/CD, is totally decentralized, which has also contributed to the low cost and simple design.

The original Ethernet LAN, as shown in Figure 2.10, was invented in the mid 1970s by Bob Metcalfe. An excellent source of online information about Ethernet is Spurgeon's Ethernet Web Site.

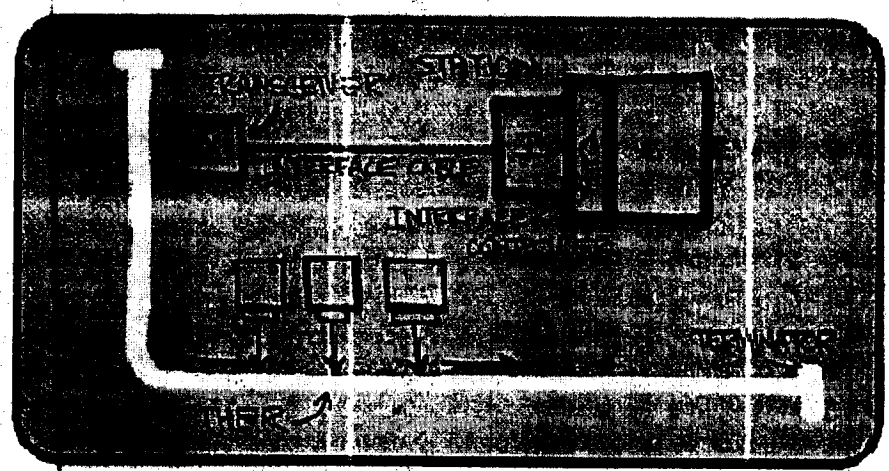


Figure 2.10: The original Metcalfe design led to the 10Base5 Ethernet standard, which included an interface cable that connected the Ethernet adapter (i.e., interface) to an external transceiver. Drawing taken from Charles Spurgeon's Ethernet Web Site.

Ethernet Basics

Today Ethernet comes in many shapes and forms. An Ethernet LAN can have a "bus topology" or a "star topology." An Ethernet LAN can run over coaxial cable, twisted-pair copper wire, or fiber optics. Furthermore, Ethernet can transmit data at different rates, specifically, at 10 Mbps, 100 Mbps and 1Gbps. But even though Ethernet comes in many flavors, all of the Ethernet technologies share a few important characteristics. Before examining the different technologies, let's first take a look at the common characteristics.

Ethernet Frame Structure

Given that there are many different Ethernet technologies on the market today, what do they have in common, what binds them together with a

common name? First and foremost is the Ethernet frame structure. All of the Ethernet technologies -- whether they use coaxial cable or copper wire, whether they run at 10 Mbps, 100 Mbps or 1 Gbps use the same frame structure.

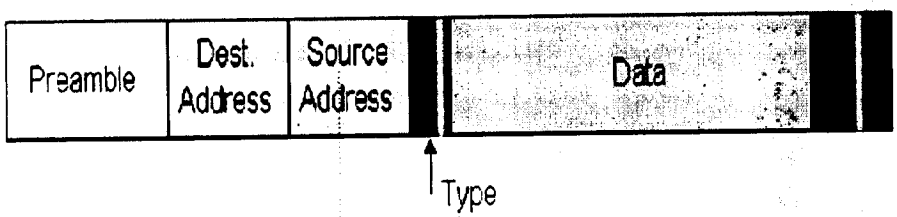


Figure 2.11 Ethernet frame structure

The Ethernet frame is shown in Figure 2.11. Once we understand the Ethernet frame, we will already know a lot about Ethernet. To put our discussion of the Ethernet frame in a tangible context, let us consider sending an IP datagram from one host to another host, with both hosts on the same Ethernet LAN. Let the sending adapter, adapter A, have physical address AA-AA-AA-AA-AA-AA and the receiving adapter, adapter B, have physical address BB-BB-BB-BB-BB-BB. The sending adapter encapsulates the IP datagram within an Ethernet frame and passes the frame to the physical layer. The receiving adapter receives the frame from the physical layer, extracts the IP datagram, and passes the IP datagram to the network layer. In this context, let us now examine the six fields of the Ethernet frame:

- **Data Field** (46 to 1500 bytes): This field carries the IP datagram. The Maximum Transfer Unit (MTU) of Ethernet is 1500 bytes. This means that if the IP datagram exceeds 1500 bytes, then the host has to fragment the datagram, as discussed in Section 4.4. The minimum size of the data field is 46 bytes. This means that if the IP datagram is less than 46 bytes, the data field has to be "stuffed" to fill it out to 46 bytes. When stuffing is used, the data passed to the network layer contains the stuffing as well as an IP datagram. The network layer uses the length field in the IP datagram header to remove the stuffing.

- **Destination Address** (6 bytes): This field contains the LAN address of the destination adapter, namely, BB-BB-BB-BB-BB-BB. When adapter B receives an Ethernet frame with destination address other than its own physical address, BB-BB-BB-BB-BB-BB, or the

LAN broadcast address, it discards the frame. Otherwise, it passes the contents of the data field to the network layer.

- **Source Address** (6 bytes): This field contains the LAN address of the adapter that transmits the frame onto the LAN, namely, AA-AA-AA-AA-AA-AA.

- **Type Field** (two bytes): The type field permits Ethernet to "multiplex" network-layer protocols. To understand this idea, we need to keep in mind that hosts can use other network-layer protocols besides IP. In fact, a given host may support multiple network layer protocols, and use different protocols for different applications. For this reason, when the Ethernet frame arrives at adapter B, adapter B needs to know to which network-layer protocol it should pass the contents of the data field. IP and other data-link layer protocols (e.g., Novell IPX or AppleTalk) each have there own, standardized type number. Furthermore, the ARP protocol (discussed in the previous section) has its own type number. Note that the type field is analogous to the protocol field in the network-layer datagram and the port number fields in the transport-layer segment; all of these fields serve to glue a protocol at one layer to a protocol at the layer above.

- **Cyclic Redundancy Check (CRC)** (4 bytes): As discussed in lesson 2, the purpose of the CRC field is to allow the receiving adapter, adapter B, to detect whether any errors have been introduced into the frame, i.e., if bits in the frame have been toggled. Causes of bit errors include attenuation in signal strength and ambient electromagnetic energy that leaks into the Ethernet cables and interface cards. Error detection is performed as follows. When host A constructs the Ethernet frame, it calculates a CRC field, which is obtained from a mapping of the other bits in frame (except for the preamble bits). When host B receives the frame, it applies the same mapping to the frame and checks to see if the result of the mapping is equal to what is in the CRC field. This operation at the receiving host is called the **CRC check**. If the CRC check fails (that is, if the result of the mapping does not equal the contents of the CRC field), then host B knows that there is an error in the frame.

- **Preamble:** (8 bytes) The Ethernet frame begins with an eight-byte preamble field. Each of the first seven bytes of the preamble is 10101010; the last byte is 10101011. The first seven bytes of the preamble serve to "wake up" the receiving adapters and to synchronize their clocks to that of the sender's clock. Why should the clocks be out of synchronization? Keep in mind that adapter A aims to transmit the frame at 10 Mbps, 100 Mbps or 1 Gbps, depending on the type of Ethernet LAN. However, because nothing is absolutely perfect, adapter A will not transmit the frame at exactly the target rate; there will always be some drift from the target rate, a drift which is not known a priori by the other adapters on the LAN. A receiving adapter can lock onto adapter A's clock by simply locking onto the bits in the first seven bytes of the preamble. The last two bits of the eighth byte of the preamble (the first two consecutive 1s) alert adapter B that the "important stuff" is about to come. When host B sees the two consecutive 1s, it knows that the next six bytes is the destination address. An adapter can tell when a frame ends by simply detecting absence of current.

An Unreliable Connectionless Service

All of the Ethernet technologies provide **connectionless service** to the network layer. That is to say, when adapter A wants to send a datagram to adapter B, adapter A encapsulates the datagram in an Ethernet frame and sends the frame into the LAN, without first "handshaking" with adapter B. This layer-2 connectionless service is analogous to IP's layer-3 datagram service and UDP's layer-4 connectionless service.

All the Ethernet technologies provide an **unreliable service** to the network layer. In particular when adapter B receives a frame from A, adapter B does not send an acknowledgment when a frame passes the CRC check (nor does it send a negative acknowledgment when a frame fails the CRC check). Adapter A hasn't the slightest idea whether a frame arrived correctly or incorrectly. When a frame fails the CRC check, adapter B simply discards the frame. This lack of reliable transport (at the link layer) helps to make Ethernet simple and cheap. But it also means that the stream of datagram's passed to the network layer can have gaps.

If there are gaps due to discarded Ethernet frames, does the application-layer protocol at host B see gaps as well? As we learned in unit-II, this solely depends on whether the application is using UDP or TCP. If the

application is using UDP, then the application-layer protocol in host B will indeed suffer from gaps in the data. On the other hand, if the application is using TCP, then TCP in host B will not acknowledge the discarded data, causing TCP in host A to retransmit. Note that when TCP retransmits data, Ethernet retransmits the data as well. But we should keep in mind that Ethernet doesn't know that it is retransmitting. Ethernet thinks it is receiving a brand new datagram with brand new data, even though this datagram contains data that has already been transmitted at least once.

Baseband Transmission and Manchester Encoding

Ethernet uses baseband transmission, that is, the adapter sends a digital signal directly into the broadcast channel. The interface card does not shift the signal into another frequency band, as do ADSL and cable modem systems. With Manchester encoding each bit contains a transition; a 1 has a transition from up to down, whereas a zero has a transition from down to up. The reason for Manchester encoding is that the clocks in the sending and receiving adapters are not perfectly synchronized. By including a transition in the middle of each bit, the receiving host can synchronize its clock to that of the sending host. Once the receiving adapter's clock is synchronized, the receiver can delineate each bit and determine whether it is a one or zero. Manchester encoding is a physical layer operation rather than a link-layer operation; however, we have briefly described it here as it is used extensively in Ethernet.

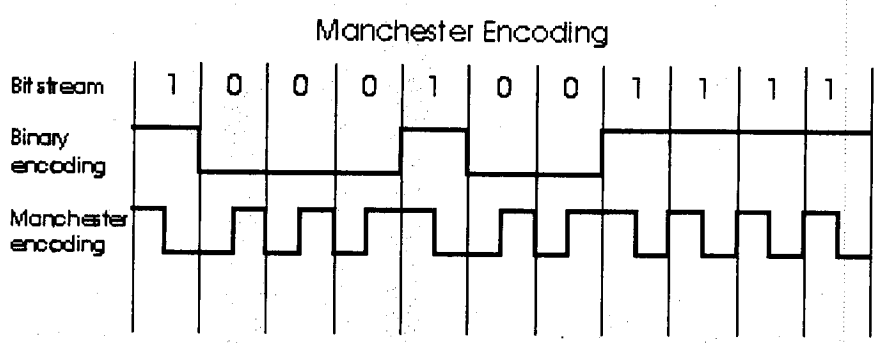


Figure 2.12 Manchester encoding

CSMA/CD: Ethernet's Multiple Access Protocol

Nodes in an Ethernet LAN are interconnected by a broadcast channel, so that when an adapter transmits a frame, all the adapters on the LAN receive the frame. As we discussed in lesson 3, Ethernet uses a CSMA/CD

multiple access algorithm. Summarizing our discussion from lesson 3, recall that CSMA/CD employs the following mechanisms:

1. An adapter may begin to transmit at any time, i.e., no slots are used.

2. An adapter never transmits a frame when it senses that some other adapter is transmitting, i.e., it uses carrier-sensing.

3. A transmitting adapter aborts its transmission as soon as it detects that another adapter is also transmitting, i.e., it uses collision detection.

4. Before attempting a retransmission, an adapter waits a random time that is typically small compared to a frame time.

These mechanisms give CSMA/CD much better performance than slotted ALOHA in a LAN environment. In fact, if the maximum propagation delay between stations is very small, the efficiency of CSMA/CD can approach 100%. But note that the second and third mechanisms listed above require each Ethernet adapter to be able to (1) sense when some other adapter is transmitting, and (2) detect a collision while it is transmitting. Ethernet adapters perform these two tasks by measuring voltage levels before and during transmission.

Each adapter runs the CSMA/CD protocol without explicit coordination with the other adapters on the Ethernet. Within a specific adapter, the CSMA/CD protocol works as follows:

1. The adapter obtains a network-layer PDU from its parent node, prepares an Ethernet frame, and puts the frame in an adapter buffer.

2. If the adapter senses that the channel is idle (i.e., there is no signal energy from the channel entering the adapter), it starts to transmit the frame. If the adapter senses that the channel is busy, it waits until it senses no signal energy (plus a few hundred microseconds) and then starts to transmit the frame.

3. While transmitting, the adapter monitors for the presence of signal energy coming from other adapters. If the adapter transmits the

entire frame without detecting signal energy from other adapters, the adapter is done with the frame.

4. If the adapter detects signal energy from other adapters while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.

5. After aborting (i.e., transmitting the jam signal), the adapter enters an **exponential backoff** phase. Specifically, when transmitting a given frame, after experiencing the nth collision in a row for this frame, the adapter chooses a value for K at random from $\{0,1,2,...,2^m - 1\}$ where m:= min(n,10). The adapter then waits K x 512 bit times and then returns to Step 2.

A few comments about the CSMA/CD protocol are certainly in order. The purpose of the jam signal is to make sure that all other transmitting adapters become aware of the collision. Let's look at an example. Suppose adapter A begins to transmit a frame, and just before A's signal reaches adapter B, adapter B begins to transmit. So B will have transmitted only a few bits when it aborts its transmission. These few bits will indeed propagate to A, but they may not constitute enough energy for A to detect the collision. To make sure that A detects the collision (so that it too can also abort), B transmits the 48-bit jam signal.

Next consider the exponential backoff algorithm. The first thing to notice here is that a bit time (i.e., the time to transmit a single bit) is very short; for a 10 Mbps Ethernet, a bit time is .1 microseconds. Now let's look at an example. Suppose that an adapter attempts for the first time to transmit a frame, and while transmitting it detects a collision. The adapter then chooses K=0 with probability .5 and chooses K=1 with probability .5. If the adapter chooses K=0, then it immediately jumps to Step 2 after transmitting the jam signal. If the adapter chooses K=1, it waits 51.2 microseconds before returning to Step 2. After a second collision, K is chosen with equal probability from {0, 1, 2, 3}. After three collisions, K is chosen with equal probability from {0, 1, 2, 3, 4, 5, 6, 7}. After ten or more collisions, K is chosen with equal probability from {0, 1, 2,..., 1023}. Thus the size of the sets from which K is chosen grows exponentially with the number of collisions (until n=10); it is for this reason that Ethernet's backoff algorithm is referred to as "exponential backoff".

The Ethernet standard imposes limits on the distance between any two nodes. These limits ensure that if adapter A chooses a lower value of K than all the other adapters involved in a collision, then adapter A will be able to transmit its frame without experiencing a new collision. We will explore this property in more detail in the homework problems.

Why use exponential backoff? Why not, for example, select K from {0,1,2,3,4,5,6,7} after every collision? The reason is that when an adapter experiences its first collision, it has no idea how many adapters are involved in the collision. If there are only a small number of colliding adapters, it makes sense to choose K from a small set of small values. On the other hand, if many adapters are involved in the collision, it makes sense to choose K from a larger, more dispersed set of values (why?). By increasing the size of the set after each collision, the adapter appropriately adapts to these different scenarios.

We also note here that each time an adapter prepares a new frame for transmission; it runs the CSMA/CD algorithm presented above. In particular, the adapter does not take into account any collisions that may have occurred in the recent past. So it is possible that an adapter with a new frame will be able to immediately sneak in a successful transmission while several other adapters are in the exponential backoff state.

Ethernet Efficiency

When only one node has a frame to send (which is typically the case), the node can transmit at the full rate of the Ethernet technology (10 Mbps, 100 Mbps, or 1Gbps). However, if many nodes have frames to transmit, the effective transmission rate of the channel can be much less. We define the **efficiency of Ethernet** to be the long-run fraction of time during which frames are being transmitted on the channel without collisions when there is a large number of active nodes, with each node having a large number of frames to send. In order to present a closed-form approximation of the efficiency of Ethernet, let t_{prop} denote the maximum time it takes signal energy to propagate between any two adapters. Let t_{trans} be the time to transmit a maximum size Ethernet frame (approximately 1.2 msecs for a 10 Mbps Ethernet). Here we simply state the following approximation:

Efficiency = $1/(1+ 5\ t_{prop}/t_{trans})$.

We see from this formula that as t_{prop} approaches 0, the efficiency approaches 1. This is intuitive because if the propagation delay is zero, colliding nodes will abort immediately without wasting the channel. Also, as t_{trans} becomes very large, efficiency approaches 1. This is also intuitive because when a frame grabs the channel, it will hold on to the channel for a very long time; thus the channel will be doing productive work most of the time.

Ethernet Technologies

The most common Ethernet technologies today are 10Base2, which uses thin coaxial cable in a bus topology and has a transmission rate of 10 Mbps; 10BaseT, which uses twisted-pair cooper wire in a star topology and has a transmission rate of 10 Mbps; 100BaseT, which typically uses twisted-pair cooper wire in a star topology and has a transmission rate of 100 Mbps; and Gigabit Ethernet, which uses both fiber and twisted-pair cooper wire and transmits at a rate of 1 Gbps. These Ethernet technologies are standardized by the IEEE 802.3 working groups. For this reason, Ethernet is often referred to as an 802.3 LAN.

Before discussing specific Ethernet technologies, we need to discuss **repeaters**, which are commonly used in LANs as well as in wide-area transport. A repeater is a physical-layer device that acts on individual bits rather than on packets. It has two or more interfaces. When a bit, representing a zero or a one, arrives from one interface, the repeater simply recreates the bit, boosts its energy strength, and transmits the bit onto all the other interfaces. Repeaters are commonly used in LANs in order to extend their geographical range. When used with Ethernet, it is important to keep in mind that repeaters do not implement carrier sensing or any other part of CSMA/CD; a repeater repeats an incoming bit on all outgoing interfaces even if there is signal energy on some of the interfaces.

10Base2 Ethernet

10Base2 is a very popular Ethernet technology. If you look at how your computer (at work or at school) is connected to the network, it is very possible you will see a 10Base2 connection. The "10" in 10Base2 stands for "10 Mbps"; the "2" stands for "200 meters", which is the approximate maximum distance between any two nodes without repeaters between them. (The actual maximum distance is 185 meters.) A 10Base2 Ethernet is shown in Figure 2.13.

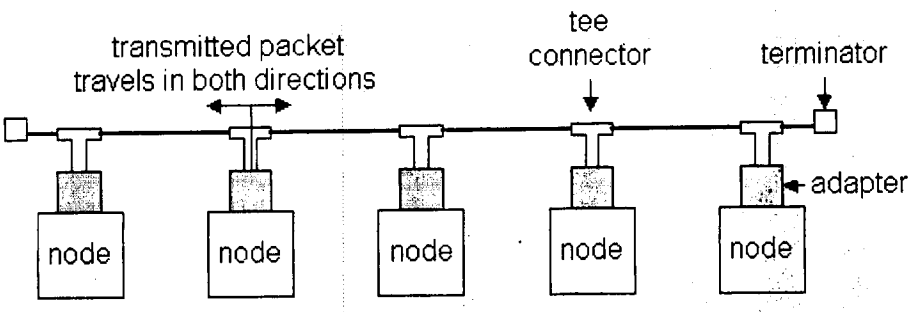


Figure 2.13 A 10Base2 Ethernet

We see from Figure 2.13 10Base2 uses a bus topology; that is, nodes are connected (through their adapters) in a linear fashion. The physical medium used to connect the nodes is **thin coaxial cable**, which is similar to what is used in cable TV, but with a thinner and lighter cable. When an adapter transmits a frame, the frame passes through a "tee connector;" two copies of the frame leave the tee connector, one copy going in one direction and one copy in the other direction. As the frames travel towards the terminators, they leave a copy at every node they pass. (More precisely, as a bit passes in front of a node, part of the energy of the bit leaks into the adapter.) When the frame finally reaches a terminator, it gets absorbed by the terminator. Note when an adapter transmits a frame, the frame is received by every other adapter on the Ethernet. Thus, 10Base2 is indeed a broadcast technology.

Suppose you want to connect a dozen PCs in your office using 10Base2 Ethernet. To do this, you would need to purchase 12 Ethernet cards with thin Ethernet ports; 12 BNC trees, which are small metallic objects that attach to the adapters (less than one dollar each); a dozen or so thin coax segments, 5-20 meters each; and two "terminators," which you put at the two ends of the bus. The cost of the whole network, including adapters, is likely to be less than the cost of a single PC! Because 10Base2 is incredibly inexpensive, it is often referred to as "cheapnet".

Without a repeater, the maximum length of a 10Base2 bus is 185 meters. If the bus becomes any longer, then signal attenuation can cause the system to malfunction. Also, without a repeater, the maximum number of nodes is 30, as each node contributes to signal attenuation. Repeaters can be used to connect 10Base2 segments in a linear fashion, with each segment having up to 30 nodes and having a length up to 185 meters. Up to four repeaters can be included in a 10Base2 Ethernet, which creates up to five "segments". Thus a 10Base2 Ethernet bus can have a total length of 985

meters and support up to 150 nodes. Note that the CSMA/CD access protocol is completely oblivious to the repeaters; if any two of 150 nodes transmit at the same time, there will be a collision.

10BaseT and 100BaseT

We discuss 10BaseT and100BaseT Ethernet together, as they are similar technologies. The most important difference between them is that 10BaseT transmits at 10 Mbps and 100BaseT Ethernet transmits at 100 Mbps. 100BaseT is also commonly called "fast Ethernet" and "100 Mbps Ethernet". 10BaseT and 100BaseT are also very popular Ethernet technologies; in fact, for new installations, 10BaseT and Ethernet are often today the technology of choice. Both 10BaseT and 100BaseT Ethernet use a star topology, as shown in Figure 2.14.

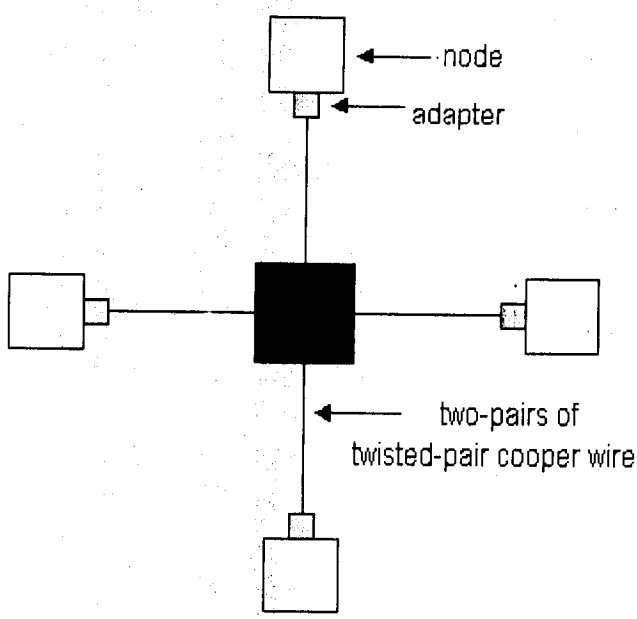


Figure 2.14 Star topology for 10BaseT and 100BaseT

In the star topology there is a central device called a **hub** (also sometimes called a concentrator.) Each adapter on each node has a direct, point-to-point connection to the hub. This connection consists of two pairs of twisted-pair cooper wire, one for transmitting and the other for receiving. At each end of the connection there is a connector that resembles the RJ-45 connector used for ordinary telephones. The "T" in 10BaseT and 100BaseT stands for "twisted pair". For both 10BaseT and 100BaseT, the maximum length of the connection between an adapter and the hub is 100 meters; the maximum length between any two nodes is 200 meters.

In essence, a hub is a repeater: when it receives a bit from an adapter, it sends the bit to all the other adapters. In this manner, each adapter can (1) sense the channel to determine if it is idle, and (2) detect a collision while it is transmitting. But hubs are popular because they also provide network management features. For example, if an adapter malfunctions and continually sends Ethernet frames (a so-called "jabbering adapter"), then in a 10Base2 Ethernet will become totally dysfunctional; none of the nodes will be able to communicate. But a 10BaseT network will continue to function, because the hub will detect the problem and internally disconnect the malfunctioning adapter. With this feature, the network administrator doesn't have to get out of bed and drive back to work in order to correct the problem for hackers who work late at night. Also, most hubs can gather information and report the information to a host that connects directly to the hub. This monitoring host provides a graphical interface that displays statistics and graphs, such as bandwidth usage, collision rates, average frame sizes, etc. Network administrators can use this information to not only debug and correct problems, but also to plan how the LAN should evolve in the future.

Many Ethernet adapters today are 10/100 Mbps adapters. This means that they can be used for both 10BaseT and 100BaseT Ethernets. 100BaseT, which typically uses category-5 twisted pair (a high-quality twisted pair with a lot of twists). Unlike the 10Base2 and 10BaseT, 100BaseT does not use Manchester encoding, but instead a more efficient encoding called 4B5B: every group of five clock periods is used to send 4 bits in order to provide enough transitions to allow clock synchronization.

We briefly mention at this point that both 10 Mbps and 100 Mbps Ethernet technologies can employ fiber links. A fiber link is often used to interconnect to hubs that are in different buildings on the same campus. Fiber is expensive because of cost of the cost of its connectors, but it has excellent noise immunity. The IEEE 802 standards permit a LAN to have a larger geographically reach when fiber is used to connect backbone nodes.

Gigabit Ethernet

Gigabit Ethernet is an extension to the highly successful 10 Mbps and 100 Mbps Ethernet standards. Offering a raw data rate of 1000 Mbps, Gigabit Ethernet maintains full compatibility with the huge installed base of Ethernet equipment. The standard for Gigabit Ethernet, referred to as IEEE 802.3z, does the following:

- Uses the standard Ethernet frame format, and is backward compatible with 10BaseT and 100BaseT technologies. This allows for easy integration of Gigabit Ethernet with the existing installed base of Ethernet equipment.

- Allows for point-to-point links as well as shared broadcast channels. Point-to-point links use switches (see in lesson 6) where as broadcast channels use hubs, as described above for 10BaseT and 100 Base-T. Un Gigabit Ethernet jargon, hubs are called "buffered distributors".

- Uses CSMA/CD for shared broadcast channels. In order to have acceptable efficiency, the maximum distance between nodes must be severely restricted.

- Allows for full-duplex operation at 1000 Mbps in both directions for point-to-point channels.

Like 10BaseT and 100BaseT, Gigabit Ethernet has a star topology with a hub or switch at its center. (Ethernet switches will be discussed in lesson 6.) Gigabit Ethernet often serves as a backbone for interconnecting multiple 10 Mbps and 100 Mbps Ethernet LANs. Initially operating over optical fiber, Gigabit Ethernet will be able to use Category 5 UTP cabling.

The Gigabit Ethernet Alliance is an open forum whose purpose is to promote industry cooperation in the development of Gigabit Ethernet.

2.7 Token Rings Technologies (802.5, FDDI)

Alongside the Ethernet, token rings are the other significant class of shared-media network. There are more different types of token rings than there are types of Ethernets; this section will discuss the type that was for years the most prevalent, known as the IBM Token Ring. Like the Xerox Ethernet, IBM's Token Ring has a nearly identical IEEE standard, known as 802.5. Where necessary, we note the differences between the IBM and 802.5 token rings.

Most of the general principles of token ring networks can he understood once the IBM and 802.5 standards have been discussed. However, the FDDI (Fiber Distributed Data Interface) standard a newer, faster type of token ring warrants some discussion, which we provide at the end of this

section. At the time of writing, yet another token ring standard, called Resilient Packet Ring or 802.17, is nearing completion.

As the name suggests, a token ring network consists of a set of nodes connected in a ring (see Figure 2.15). Data always flows in a particular direction around the ring, with each node receiving frames from its upstream neighbor and then forwarding them to its downstream neighbor. This ring-based topology is in contrast to the Ethernet's bus topology. Like the Ethernet, however, the ring is viewed as a single shared medium; it does not behave as a collection of independent point-to-point links that just happen to be configured in a loop. Thus, a token ring shares two key features with an Ethernet: First, it involves a distributed algorithm that controls when each node is allowed to transmit, and second, all nodes see all frames, with the node identified in the frame header as the destination saving a copy of the frame as it flows past.

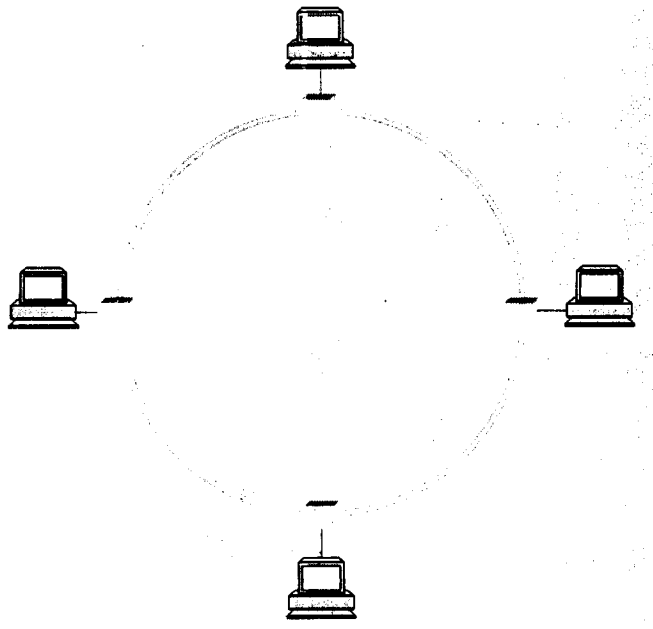


Figure 2.15 Token ring network

The word "token" in token ring comes from the way access to the shared ring is managed. The idea is that a token, which is really just a special sequence of bits, circulates around the ring; each node receives and then forwards the token. When a node that has a frame to transmit sees the token, it takes the token off the ring (i.e., it does not forward the special bit pattern) and instead inserts its frame into the ring. Each node along the

way simply forwards the frame, with the destination node saving a copy and forwarding the message onto the next node on the ring. When the frame makes its way hack around to the sender, this node strips its frame off the ring (rather than continuing to forward it) and reinserts the token. In this way, some node downstream will have the opportunity to transmit a frame. The media access algorithm is fair in the sense that as the token circulates around the ring each node gets a chance to transmit. Nodes are serviced in a round-robin fashion.

Physical Properties

One of the first things you might worry about with a ring topology is that any link or node failure would render the whole network useless. This problem is addressed by connecting each station into the ring using an electromechanical relay. As long as the station is healthy, the relay is open and the station is included in the ring, if the station stops providing power, the relay closes and the ring automatically bypasses the station. This is illustrated in Figure 2.16.

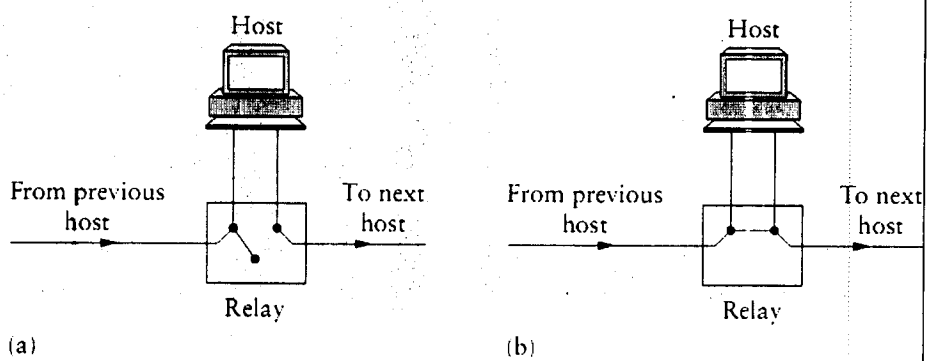


Figure 2.16 Relay used on a token ring: (a) relay open host active; (b) relay closed host bypassed

Several of these relays are usually packed into a single box, known as a multi- station access unit (MSAU). This has the interesting effect of making a token ring actually look more like a star topology, as shown in Figure 2.17. It also makes it very easy to add stations to and remove stations from the network, since they can just he plugged into or unplugged from the nearest MSAU, while the overall wiring of the network can he left unchanged. One of the small differences between the IBM Token Ring specification and 802.5 is that the former actually requires the use of MSAUs, while the latter does not in practice MSAUs are almost always

used because of the need for robustness and ease of station addition and removal.

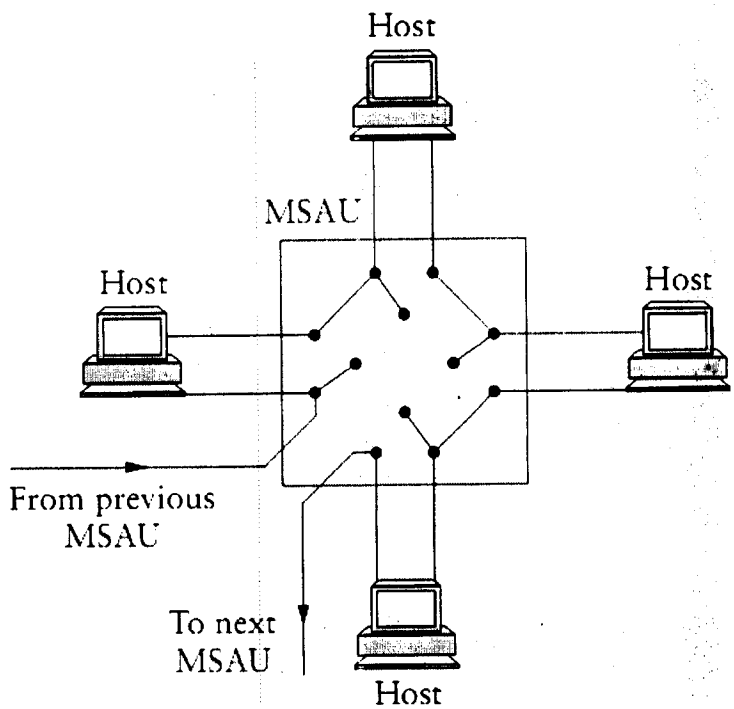


Figure 2.17 Multistation access unit

There are a few other physical details to know about 802.5 and IBM Token Rings. The data rate may be either 4 Mbps or 16 Mbps. The encoding of bits uses differential Manchester encoding, as described in Section 2.2. IBM Token Rings may have up to 260 stations per ring, while 802.5 sets the limit at 250. The physical medium is twisted pair for IBM, but is not specified in 802.5.

Token Ring Media Access Control

It is now time to look a little more closely at how the MAC protocol operates on a token ring. The network adaptor for a token ring contains a receiver, a transmitter, and one or more bits of data storage between them. When none of the stations connected to the ring has anything to send, the token circulates around the ring. Obviously, the ring has to have enough "storage capacity" to hold an entire token. For example, the 802.5 token is 24 bits long, if every station could hold only 1 bit (as is the norm for 802.5 networks), and the stations were close enough together that the time for a bit to propagate from one station to another was negligible, we would need

to have at least 24 stations on the ring before it would operate correctly. This situation is avoided by having one designated station, called the monitor add some additional bits of delay to the ring if necessary. The operation of the monitor is described in more detail below.

As the token circulates around the ring, any station that has data to send may "seize" the token, that is, drain it off the ring and begin sending data. In 802.5 networks, the seizing process involves simply modifying 1 bit in the second byte token; the first 2 bytes of the modified token now become the preamble for the subsequent data packet. Once a station has the token, it is allowed to send one or more packets exactly how many more depends on some factors described below.

Each transmitted packet contains the destination address of the intended receiver; it may also contain a multicast (or broadcast) address if it is intended to reach more than one (or all) receivers. As the packet flows past each node on the ring, each node looks inside the packet to see if it is the intended recipient. If so, it copies the packet into a buffer as it flows through the network adaptor, hut it does not remove the packet from the ring. The sending station has the responsibility of removing the packet from the ring. For any packet that is longer than the number of bits that can be stored in the ring, the sending station will be draining the first part of the packet from the ring while still transmitting the latter part.

One issue we must address is how much data a given node is allowed to transmit each time it possesses the token, or said another way, how long a given node is allowed to hold the token. We call this the token holding time (THT). If we assume that most nodes on the network do not have data to send at any given time a reasonable assumption, and certainly one that the Ethernet takes advantage of then we could make a case for letting a node that possesses the token transmit as much data as it has before passing the token on to the next node. This would mean setting the THT to infinity, it would he silly in this case to limit a node to sending a single message and to force it to wait until the token circulates all the way around the ring before getting a chance to send another message. Of course, "as much data as it has" would he dangerous because a single station could keep the token for an arbitrarily long time, but we could certainly set the THT to significantly more than the time to send one packet.

It is easy to see that the more bytes a node can send each time it has the token, the better the utilization of the ring you can achieve in the situation in

which only a single node has data to send. The downside, of course, is that this strategy does not work well when multiple nodes have data to send it favors nodes that have a lot of data to send over nodes that have only a small message to send, even when it is important to get this small message delivered as soon as possible. The situation is analogous to finding yourself in line at the bank behind a customer who is taking out a car loan, even though you simply want to cash a check. In 802.5 networks, the default THT is 10ms.

There is a little subtlety to the use of the THT. Before putting each packet onto the ring, the station must check that the amount of time it would take to transmit the packet would not cause it to exceed the token holding time. This means keeping track of how long it has already held the token, and looking at the length of the next packet that it wants to send.

From the token holding time we can derive another useful quantity, the token rotation time (TRT), which is the amount of time it takes a token to traverse the ring as viewed by a given node. It is easy to see that

TRT≤ Active Nodes x THT + Ring Latency

where Ring Latency denotes how long it takes the token to circulate around the ring 'when no one has data to send, and Active Nodes denotes the number of nodes that have data to transmit.

The 802.5 protocol provides a form of reliable delivery using 2 bits in the packet trailer, the A and C bits. These are both 0 initially. When a station sees a frame for which it is the intended recipient, it sets the A bit in the frame. When it copies the frame into its adaptor, it sets the C bit. if the sending station sees the frame come hack over the ring with the A bit still 0, it knows that the intended recipient is not functioning or absent. If the A bit is set but not the C bit, this implies that for some reason (e.g., lack of buffer space) the destination could not accept the frame. Thus, the frame might reasonably he retransmitted later in the hope that buffer space had become available.

Another detail of the 802.5 protocol concerns the support of different levels of priority. The token contains a 3-bit priority field, so we can think of the token having a certain priority n at any time. Each device that wants to send a packet assigns a priority to that packet, and the device can only seize the token to transmit a packet if the packet's priority is at least as

great as the token's. The priority of the token have changes over time due to the use of three reservation bits in the frame header. For example, a station X waiting to send a priority n packet may set these bits to n if it sees a data frame going past and the bits have not already been set to a higher value. This causes the station that currently holds the token to elevate its priority to ii when it releases it. Station X is responsible for lowering the token priority to its old value when it is done.

Note that this is a strict priority scheme, in the sense that no lower-priority packets get sent when higher-priority packets are waiting. This may cause lower-priority packets to he locked out of the ring for extended periods if there is a sufficient supply of high-priority packets.

One final issue will complete our discussion of the MAC protocol, which is the matter of exactly when the sending node releases the token. As illustrated in Figure 2.18, the sender can insert the token back onto the ring immediately following its frame (this is called earl)' release) or after the frame it transmits has gone all the way around the ring and been removed (this is called delayed release). Clearly, early release allows better bandwidth utilization, especially on large rings. 802.5 originally used delayed token release, hut support for early release was subsequently added.

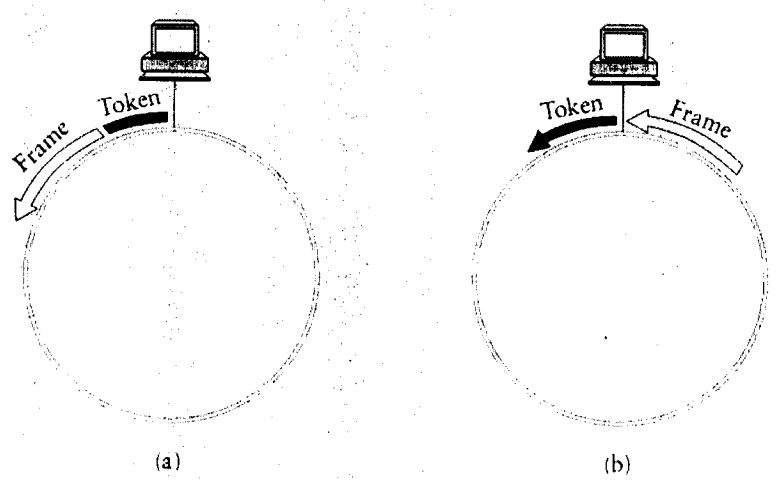


Figure 2.18 Token release: (a) early versus (b) delayed

Token Ring Maintenance

As we noted above, token rings have a designated monitor station. The monitor's job is to ensure the health of the ring. Any station on the ring can

become the monitor, and there are defined procedures by which the monitor is elected when the ring is first connected or on the failure of the current monitor. A healthy monitor periodically announces its presence with a special control message; if a station fails to see such a message for some period of time, it will assume that the monitor has failed and will try to become the monitor. The procedures for electing a monitor are the same whether the ring has just come up or the active monitor has just failed.

When a station decides that a new monitor is needed, it transmits a "claim token" frame, announcing its intent to become the new monitor. if that token circulates back to the sender, it can assume that it is OK for it to become the monitor. If some other station is also trying to become the monitor at the same instant, the sender might see a claim token message from that other station first. In this case, it will be necessary to break the tie using some well-defined rule like "highest address wins."

Once the monitor is agreed upon, it plays a number of roles. We have already seen that it may need to insert additional delay into the ring. It is also responsible for making sure that there is always a token somewhere in the ring, either circulating or currently held by a station. It should he clear that a token may vanish for several reasons, such as a bit error, or a crash on the part of a station that was holding it. To detect a missing token, the monitor watches for a passing token and maintains a timer equal to the maximum possible token rotation time. This interval equals

Num Stations x THT + Ring Latency

where Num Stations is the number of stations on the ring, and Ring Latency is the total propagation delay of the ring. If the timer expires without the monitor seeing a token, it creates a new one.

The monitor also checks for corrupted or orphaned frames. The former have checksum errors or invalid formats, and without monitor intervention, they could circulate forever on the ring. The monitor drains them off the ring before reinserting the token. An orphaned frame is one that was transmitted correctly onto the ring hut whose "parent" died; that is, the sending station went down before it could remove the frame from the ring. These are detected using another header bit, the "monitor" bit. This is 0 on transmission and set to 1 the first time the packet passes the monitor. if the monitor sees a packet with this bit set, it knows the packet is going by for the second time and it drains the packet off the ring.

One additional ring maintenance function is the detection of dead stations. The relays in the MSAU can automatically bypass a station that has been disconnected or powered down, but may not detect more subtle failures. If any station suspects a failure on the ring, it can send a beacon frame to the suspect destination. Based on how far this frame gets, the status of the ring can he established, and malfunctioning stations can he bypassed by the relays in the MSAU.

Frame Format

We are now ready to define the 802.5 frame format, which is depicted in Figure 2.19. As noted above, 802.5 use differential Manchester encoding. This fact is used by the frame format, which uses "illegal" Manchester codes in the start and end delimiters.

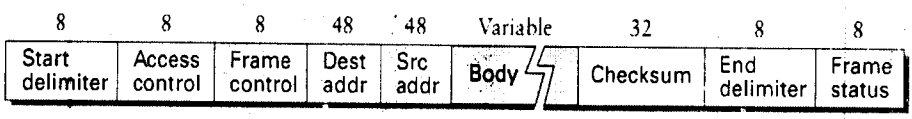


Figure 2.19 802.5 token ring frame format

After the start delimiter comes the access control byte, which includes the frame priority and the reservation priority mentioned above. The frame control byte is a demux key that identifies the higher-layer protocol.

Similar to the Ethernet, 802.5 addresses are 48 bits long. The standard actually allows for smaller 16-bit addresses, hut 48-bit addresses are. typically used. When 48- bit addresses are used, they are interpreted in exactly the same way as on an Ethernet. The frame also includes a 32-bit CRC. This is followed by the frame status byte, which includes the A and C bits for reliable delivery.

FDDI

In many respects, FDDI is similar to 802.5 and IBM Token Rings. However, there are significant differences some arising because it runs on fiber, not copper, and some arising from innovations that were made subsequent to the invention of the IBM Token Ring. We discuss some of the significant differences below.

Physical Properties

Unlike 802.5 networks, an FDD1 network consists of a dual ring two independent rings that transmit data in opposite directions, as illustrated in Figure 2.20(a). The second ring is not used during normal operation hut instead comes into play only if the primary ring fails, as depicted in Figure 2.20(b). That is, the ring loops hack on the secondary fiber to form a complete ring, and as a consequence, an FDDI network is able to tolerate a single break in the cable or the failure of one station.

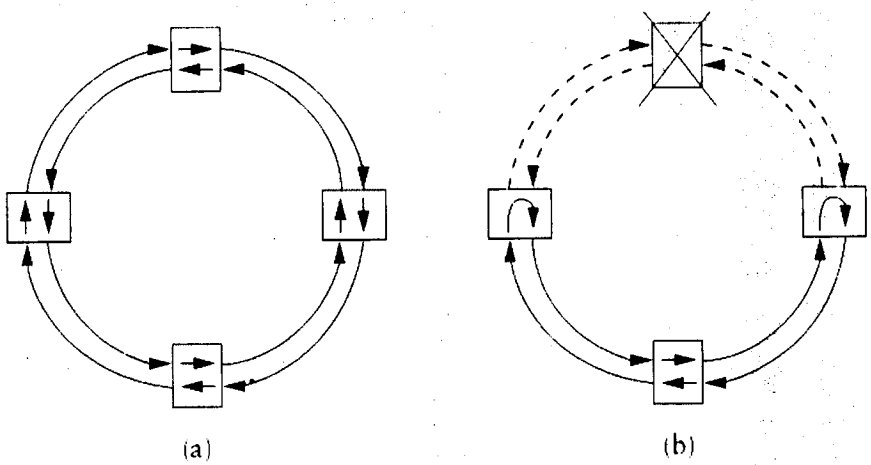


Figure 2.20 Dual-fiber ring: (a) normal operation; (b) failure of the primary ring.

Because of the expense of the dual-ring configuration, FDDI allows nodes to attach to the network by means of a single cable. Such nodes are called single attachment stations (SAS); their dual-connected counterparts are called, not surprisingly, dual attachment stations (DAS). A concentrator is used to attach several SAS to the dual ring as illustrated in Figure 2.21. Notice how the single-cable (two-fiber) connection into an SAS forms a connected piece of the ring. Should this SAS fail, the concentrator detects this situation and uses an optical bypass to isolate the failed SAS, thereby keeping the ring connected. This is analogous to the relays inside MSAUs used in 802.5 rings. Note that in this illustration, the second (backup) ring is denoted with a dotted line.

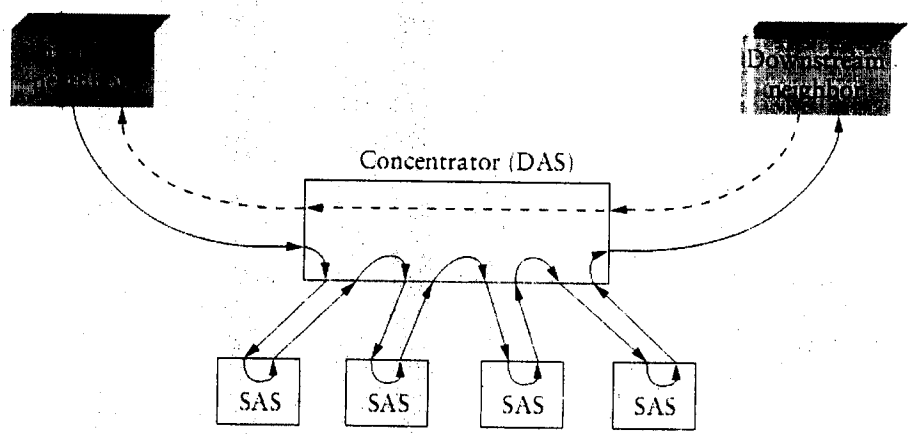


Figure 2.21 SASs connected to a concentrator

As in 802.5, each network adaptor holds some number of bits between its input and output interfaces. Unlike 802.5, however, the buffer cans be of different sizes in different stations, although never less than 9 bits nor more than 80 bits. It is also possible for a station to start transmitting bits out of this buffer before it is full. Of course, the total time it takes for a token to pass around the network is a function of the size of these buffers. For example, because FDDI is a 100-Mhps network, it has a 10-nanosecond (ns) bit rime (each bit is 10 ns wide). if each station implements a 10-bit buffer and waits for the buffer to be half full before starting to transmit, then each station introduces a 5 x 10 ns = 50-ns delay into the total ring rotation time.

FDDI has other physical characteristics. For example, the standard limits a single network to at most 500 stations (hosts), with a maximum distance of 2 km between any pair of stations. Overall, the network is limited to a total of 200 km of fiber, which means that, because of the dual nature of the ring, the total amount of cable connecting all stations is limited to 100 km. Also, although the "F" in FDDI implies that optical fiber serves as the underlying physical medium, the standard has been defined to run over a number of different physical media, including coax and twisted pair. Of course, you still have to be careful about the total distance covered by the ring. As we will see below, the amount of time it takes the token to traverse the network plays an important role in the access control algorithm.

Timed Token Algorithm

The rules governing token holding times are a little more complex in FDDI than in 802.5. The THT for each node is defined as before and is

configured to some suitable value. In addition, to ensure that a given node has the opportunity to transmit within a certain amount of time that is, to put an upper hound on the TRT observed by any node we define a target token rotation time (TTRT), and all nodes agree to live within the limits of the TTRT. (How the nodes agree to a particular TTRT is described in the next subsection.) Specifically, each node measures the time between successive arrivals of the token. We call this the node's measured TRT. If this measured TRT is greater than the agreed-upon TTRT, then the token is late, and the node does not transmit any data. If this measured TRT is less than the TTRT then the token is early, and the node is allowed to hold the token for the difference between TTRT and the measured TRT.

Although it may seem that we are now done, the algorithm we have just developed does not ensure that a node concerned with sending a frame with a hounded delay will actually he able to do so. The problem is that a node with lots of data to send has the opportunity, upon seeing an early token, to hold the token for so long that by the time a downstream node gets the token, its measured TRT is equal to or exceeds the TTRT, meaning that it still cannot transmit its frame. To account for this possibility, FDDI defines two classes of traffic: synchronous and asynchronous.3 When a node receives a token, it is always allowed to send synchronous data, without regard for whether the token is early or late. In contrast, a node can send asynchronous traffic only when the token is early.

Note that the terms synchronous and asynchronous are somewhat misleading. By synchronous, FDDI means that the traffic is delay sensitive. For example, you would send voice or video as synchronous traffic on an FDDI network. In contrast, asynchronous means that the application is more interested in throughput than delay. A file transfer application would he asynchronous FDDI traffic.

Are we done yet? Not quite. Because synchronous traffic can transmit without regard to whether the token is early or late, it would seem that if each node had a sizable amount of synchronous data to send, then the target rotation time would again he meaningless. To account for this, the total amount of synchronous data that can he sent during one token rotation is also hounded by TTRT. This means that in the worst case, the nodes with asynchronous traffic first use up one TTRT's worth of time, and then the nodes with synchronous data consume another TTRT's worth of time, meaning that it is possible for the measured TRT at any given node to he as much as 2 x TTRT. Note that if the synchronous traffic has already

consumed one TTRT's worth of time, then the nodes with asynchronous traffic will not send any data because the token will he late. Thus, while it is possible for a single rotation of the token to take as long as 2 x TTRT, it is not possible to have back-to-back rotations that take 2 x TTRT amount of time.

One final detail concerns precisely how a node determines if it can send asynchronous traffic. As stated above, a node sends if the measured TRT is less than the YTRT. The question then arises: What if the measured TRT is less than the TTRT, hut by such a small amount that it's not possible to send the full message without exceeding the TTRT? The answer is that the node is allowed to send in this case. As a consequence, the measured TRT is actually bounded by YTRT plus the time it takes to send a full FDDI frame.

Token Maintenance

The FDDI mechanisms for ensuring that a valid token is always in circulation are also different from those in 802.5, as they are intertwined with the process of setting the TTRT. First all nodes on an FDDI ring monitor the ring to be sure that the token has not been lost. Observe that in a correctly functioning ring, each node should see a valid transmission either a data frame or the token every so often. The greatest idle time between valid transmissions that a given node should experience is equal to the ring latency plus the time it takes to transmit a full frame, which on a maximally sized ring is a little less than 2.5ms. Therefore, each node sets a timer event those fires after 2.5ms. If this timer expires, the node suspects that something has gone wrong and transmits a "claim" frame. Every time a valid transmission is received, however, the node resets the timer back to 2.5ms.

The claim frames in FDDI differ from those in 802.5 because they contain the node's bid for the TIRT, that is, the token rotation time that the node needs so that the applications running on the node can meet their timing constraints. A node can send a claim frame without holding the token and typically does so whenever it suspects a failure or when it first joins the network. If this claim frame makes it all the way around the ring, then the sender removes it, knowing that its TIRT hid was the lowest. That node now holds the token that it is responsible for inserting a valid token on the ring and may proceed with the normal token algorithm.

When a node receives a claim frame, it checks to see if the TTRT hid in the frame is less than its own. If it is, then the node resets its local definition of the TTRT to that contained in the claim frame and forwards the frame to the next node. If the hid TTRT is greater than that node's minimum required TTRT, then the claim frame is removed from the ring and the node enters the bidding process by putting its own claim frame on the ring. Should the hid TTRT he equal to the node's required TTRT, the node compares the address of the claim frame's sender with its own and the higher address wins. Thus, if a claim frame makes it all the way back around to the original sender, that node knows that it is the only active bidder and that it can safely claim the token. At the same time, all nodes are now in agreement about the FRT that will he short enough to keep all nodes happy.

Frame Format

The FDDI frame format, depicted in Figure 2.22, differs in very few ways from that for 802.5. Because FDDI uses 4B/5B encoding instead of Manchester, it uses 4BISB control symbols rather than illegal Manchester symbols in the start- and end-of-frame markers. The other significant differences are the presence of a bit in the header to distinguish synchronous from asynchronous traffic, and the lack of the access control bits of 802.5.

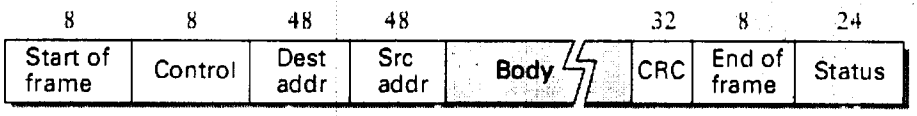


Figure 2.22 FDDI frame format

2.8 Metropolitan Area Networks (MAN)

MAN is designed to extend over the entire city. It may be a single network as a cable TV network or it may be means of connecting a number of LANs into a larger network so that resources may be shared as shown in Figure 2.23. For example, a company can use a MAN to connect the LANs in all its offices in a city. MAN is wholly owned and operated by a private company or may be a service provided by a public company.

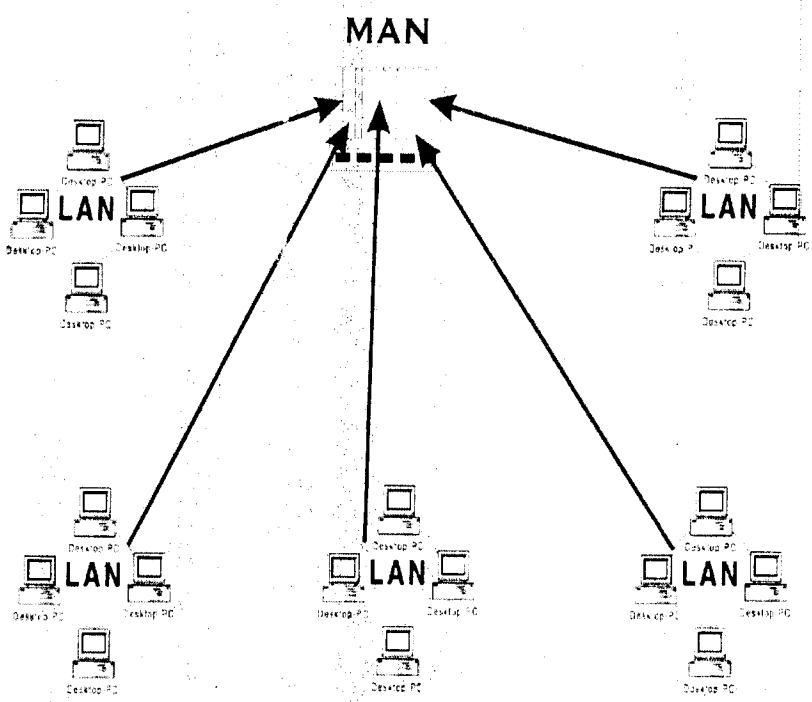


Figure 2.23 Metropolitan Area Networks (MAN)

The main reason for distinguishing MANs as a special category is that a standard has been adopted for them. It is **DQDB** (Distributed Queue Dual Bus) or IEEE 802.6.

2.9 Wide Area Network

A wide area network (WAN) provides long-distance transmission of data, image, audio, and video information over large geographic areas that may comprise a country, a continent, or even the whole world. A WAN can be as complex as the backbones that connect the internet or as simple as a dial-up line that connects a home computer to the Internet. We normally refer to the first as a switched WAN and to the second as a point-to-point WAN. The switched WAN connects the end systems, which usually comprise a router (internetworking connecting device) that connects to another LAN or WAN. The point-to-point WAN is normally a line leased from a telephone or cable TV provider that connects a home computer or a small LAN to an Internet service provider (ISP). This type of WAN is often used to provide Internet access. A network designed to provide connectivity between end users X.25 is being gradually replaced by a high-speed, more efficient network called Frame Relay. A good example of a switched WAN

is the asynchronous transfer mode (ATM) network, which is a network with fixed-size data unit packets called cells.

Frame Relay

Frame Relay is a virtual-circuit wide-area network that was designed in response to demands for a new type of WAN in the late 1980s and early I 990s.

1. Prior to Frame Relay, some organizations were using a virtual-circuit switching network called X.25 that performed switching at the network layer. For example, the Internet, which needs wide-area networks to carry its packets from one place to another, used X.25. And X.25 is still being used by the Internet, but it is being replaced by other WANs. However, X.25 has several drawbacks:

   a. X.25 has a low 64-kbps data rate. By the 1990s, there was a need for higher data-rate WANs.

   b. X.25 has extensive flow and error control at both the data link layer and the network layer. This was so because X.25 was designed in the l970s, when the available transmission media were more prone to errors. Flow and error control at both layers create a large overhead and slow down transmissions. X.25 1quires acknowledgments for both data link layer frames and network layer packets that are sent between nodes and between source and destination.

   c. Originally X.25 was signed for private use, not for the Internet. X.25 has its own network layer. This means that the user's data are encapsulated in the network layer packets of X.25. The Internet, however, has its own network layer, which means ifthe Internet wants to use X.25, the Internet must deliver its network layer packet, called a datagram, to X.25 for encapsulation in the X.25 packet. This doubles the overhead.

2. Disappointed with X.25, some organizations started their own private WAN by leasing T-l or T-3 lines from public service providers. This approach also has some drawbacks.

   a. If an organization has n branches spread over an area, it needs n(n - 1)2 T-1 or T-3 lines. The organization pays for all these lines

although it may use the lines only 10 percent of the time. This can be very costly:

b. The services provided by T-1 and T-3 lines assume that the user has fixed-rate data all the time. For example, a T-1 line is designed for a user who wants to use the line at a consistent 1.544Mbps. This type of service is not suitable for the many users today that need to send bursty data. For example, a user may want to send data at 6 Mbps for 2 s, 0 Mbps (nothing) for 7 s and 3.44 Mbps for 1 s for a total of 15.44Mbits during a period of 10 s. Although the average data rate is still 1.544Mbps, the T-1 line cannot accept this type of demand because it is designed for fixed-rate data, not bursty data. Bursty data require what is called bandwidth on demand. The user needs different bandwidth allocations at different times.

In response to the above drawbacks, Frame Relay was designed. Frame Relay is a wide- area network with the following features:

1. Frame Relay operates at a higher speed (1 .544 Mbps and recently 44.3 76 Mbps). This means that it can easily be used instead of a mesh of T-1 or T-3 lines.

2. Frame Relay operates in just the physical and data link layers. This means it can easily be used as a backbone network to provide services to protocols that already have a network layer protocol, such as the Internet.

3. Frame Relay allows bursty data.

4. Frame Relay allows a frame size of 9000 bytes, which can accommodate all local- area network frame sizes.

5. Frame Relay is less expensive than other traditional WANs.

6. Frame Relay has error detection at the data link layer only. There is no flow control or error control. There is not even a 1transmission policy if a frame is damaged; it is silently dropped. Frame Relay was designed in this way to provide fast transmission capability for more reliable media and for those protocols that have flow and error control at the higher layers.

SMDS

Switched Multimegabit Data Service (SMDS). A service supporting LAN-to-WAN connectivity offered by some telephone companies. SMDS is public, packet switched network that is primarily used for sending large amount of data sporadically. SMDS resembles the Public Switched Telephone Network (PSTN) in that companies are assigned address that is unique, 10-digit number. Unlike Frame Relay, there are no virtual circuits. The path through the network is completely dynamic. So that any company can send to any other company without configuring circuits. Because SMDS packets contain up to 7168, bytes most common LAN frames can be sent without fragmentation

ISDN

Integrated Services Digital Network (ISDN), a digital communication service offered by telephone carriers and standardized by ITU-T. ISDN combines voice connection and digital data services in a single physical medium. An ISDN connection includes two 64-Kbps channels, one that can be used to transmit data and another that can be used for digitized voice. (A device that encodes analog voice into a digital ISDN link is called a CODEC, for coder/decoder.) When the voice channel is not in use, it can be combined with the data channel to support up to 128 Kbps of data bandwidth.

For many years ISDN was viewed as the future for modest bandwidth into the home. ISDN has now been largely overtaken, however, by two newer technologies: DSL (digital subscriber line) and cable modems. The former is actually a collection of technologies that are able to transmit data at high speeds over the standard twisted pair lines that currently come into most homes in the United States (and many other places). The one in most widespread use today is ADSL (asymmetric digital subscriber line). As its name implies, ADSL provides a different bandwidth from the subscriber to the telephone company's central office (upstream) than it does from the central office to the subscriber (downstream). The exact bandwidth depends on the length of the line running from the subscriber to the central office.

SONETT

The Synchronous Optical Network (SONET) standard. For lack of a widely accepted generic term, we refer to this approach simply as clock-based

framing. SONET was first proposed by Bell Communications Research (Bellcore), and then developed under the American National Standards Institute (ANSI) for digital transmission over optical fiber; it has since been adopted by the ITU-T. Who standardized that and when is not the interesting issue, though. The thing to remember about SONET is that it is the dominant standard for long-distance transmission of data over optical networks.

An important point to make about SONET before we go any further is that the full specification is substantially larger than this hook. Thus, the following discussion will necessarily cover only the high points of the standard. Also, SONET addresses both the framing problem and the encoding problem. It also addresses a problem that is very important for phone companies the multiplexing of several low-speed links onto one high-speed link. We begin with framing and discuss the other issues following.

As with the previously discussed framing schemes, a SONET frame has some special information that tells the receiver where the frame starts and ends. However, that is about as far as the similarities go. Notably, no bit stuffing is used, so that a frame's length does not depend on the data being sent. So the question to ask is, how does the receiver know where each frame starts and ends? We consider this question for the lowest-speed SONET link, which is known as STS-1 and runs at 51.84 Mbps. An STS-1 frame is shown in Figure 2.24. It is arranged as nine rows of 90 bytes each, and the first 3 bytes of each row are overhead, with the rest being available for data that is being transmitted over the link. The first 2 bytes of the frame contain a special bit pattern, and it is these bytes that enable the receiver to determine where the frame starts. However, since bit stuffing is not used, there is no reason why this pattern will not occasionally turn up in the payload portion of the frame. To guard against this, the receiver looks for the special bit pattern consistently, hoping to see it appearing once every 810 bytes, since each frame is 9 x 90 = 810 bytes long. When the special pattern turns up in the right place enough times, the receiver concludes that it is in sync and can then interpret the frame correctly.

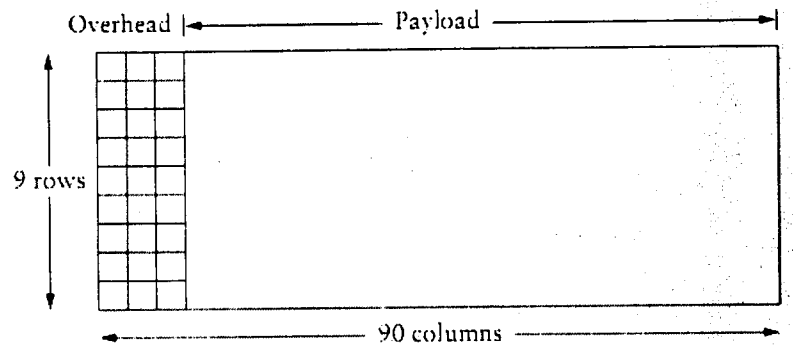


Figure 2.24 A SONET STS-1frame

One of the things we are not describing due to the complexity of SONET is the detailed use of all the other overhead bytes. Part of this complexity can be attributed to the fact that SONET runs across the carrier's optical network, not just over a single link. (Recall that we are glossing over the fact that the carriers implement a network, and we are instead focusing on the fact that we can lease a SONET link from them and then use this link to build our own packet-switched network.) Additional complexity comes from the fact that SONET provides a considerably richer set of services than just data transfer. For example, 64 Kbps of a SONET link's capacity is set aside for a voice channel that is used for maintenance.

The overhead bytes of a SONET frame are encoded using NRZ, the simple encoding described in the previous section where is are high and Os are low. However, to ensure that there are plenty of transitions to allow the receiver to recover the sender's clock, the payload bytes are scrambled. This is done by calculating the exclusive-OR (XOR) of the data to be transmitted and by the use of a well-known bit pattern. The bit pattern, which is 127 bits long, has plenty of transitions from 1 to 0, so that XORing it with the transmitted data is Likely to yield a signal with enough transitions to enable clock recovery.

Point-to-Point Protocol (PPP)

The more recent Point-to-Point Protocol (PPP), which is commonly run over dial- up modem links. The format for a PPP frame is given in Figure 2.24. The special start-of-text character, denoted as the Flag field in Figure 2.25, is 01111110. The Address and Control fields usually contain default values, and so are uninteresting. The Protocol field is used for demultiplexing:

it identifies the high-level protocol such as IP or IPX (an IP-like protocol developed by Novell). The frame payload size can he negotiated, hut it is 1500 bytes by default. The Checksum field is either 2 (by default) or 4 bytes long.

The PPP frame format is unusual in that several of the field sizes are negotiated rather than fixed. This negotiation is conducted by a protocol called LCP (Link Control Protocol). PPP and LCP work in tandem: LCP sends control messages encapsulated in PPP frames such messages are denoted by an LCP identifier in the PPP Protocol field and then turns around and changes PPP's frame format based on the information contained in those control messages. LCP is also involved in establishing a link between two peers when both sides detect the carrier signal.

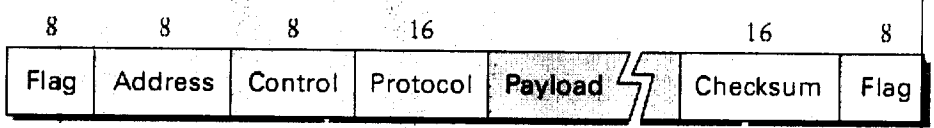


Figure 2.25 PPP frame format

High-Level Data Link Control (HDLC)

Unlike these byte-oriented protocols, a bit- oriented protocol is not concerned with byte boundaries it simply views the frame as a collection of bits. These bits might come from some character set, such as ASCII, they might be pixel values in an image, or they could he instructions and operands from an executable file. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. In the following discussion, we use HDLC as an example; its frame format is given in Figure 2.26.

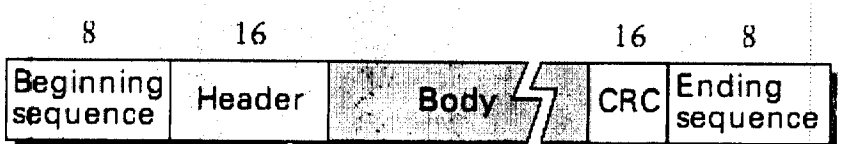


Figure 2.26 HDLC frame format

HDLC denotes both the beginning and the end of a frame with the distinguished bit sequence 01111110. This sequence is also transmitted during any times that the link is idle so that the sender and receiver can

keep their clocks synchronized. In this way, both protocols essentially use the sentinel approach. Because this sequence might appear anywhere in the body of the frame in fact, the bits 01111110 might cross byte boundaries bit-oriented protocols use the analog of the DLE character, a technique known as bit stuffing.

Bit stuffing in the HDLC protocol works as follows. On the sending side, any time five consecutive is have been transmitted from the body of the message (i.e., excluding when the sender is trying to transmit the distinguished 01111110 sequence), the sender inserts a 0 before transmitting the next bit. On the receiving side, should five consecutive is arrive, the receiver makes its decision based on the next bit it sees (i.e., the bit following the five is). If the next bit is a 0, it must have been stuffed, and so the receiver removes it. If the next bit is a i, then one of two things is true: Either this is the end-of-frame marker or an error has been introduced into the bit stream. By looking at the next bit, the receiver can distinguish between these two cases: If it sees a 0 (i.e., the last eight bits it has looked at are 01111110), then it is the end-of- frame marker; if it sees a i (i.e., the last eight bits it has looked at are 01111111), then there must have been an error and the whole frame is discarded. In the latter case, the receiver has to wait for the next 01111110 before it can start receiving again, and as a consequence, there is the potential that the receiver will fail to receive two consecutive frames. Obviously, there are still ways that framing errors can go undetected, such as when an entire spurious end-of-frame pattern is generated by errors, hut these failures are relatively unlikely.

An interesting characteristic of bit stuffing, as well as character stuffing, is that the size of a frame is dependent on the data that is being sent in the payload of the frame. It is in fact not possible to make all frames exactly the same size, given that the data that might be carried in any frame is arbitrary. (To convince yourself of this, consider what happens if the last byte of a frame's body is the ETX character.) A form of framing that ensures that all frames are the same size is described in the next subsection.

Logical Link Control (LLC)

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

The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LAN s. A single LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent. Figure 2.27 shows one single LLC protocol serving several MAC protocols.

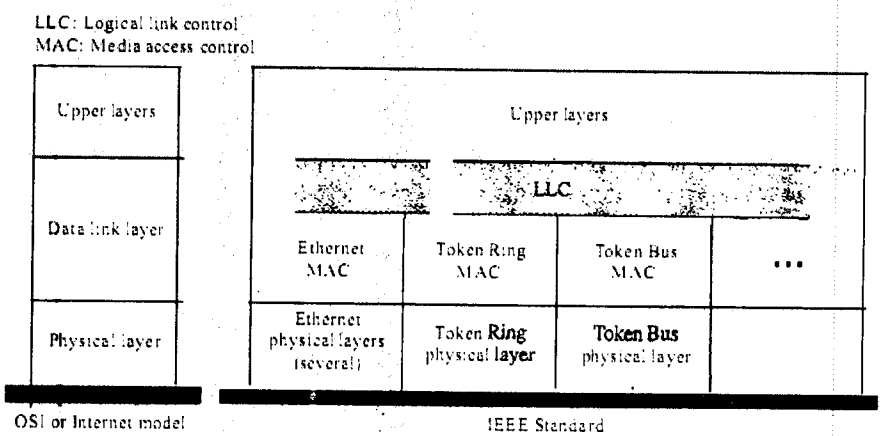


Figure 2.27 IEEE standard or LANs

Framing LLC defines a protocol data unit (PDL1) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the tipper-layer protocol at the source and destination that uses LLC. These fields are called the destination service access point (DSAP) and the source service access point (SSAP). The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. in other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sub layer, as shown in Figure 2.28.

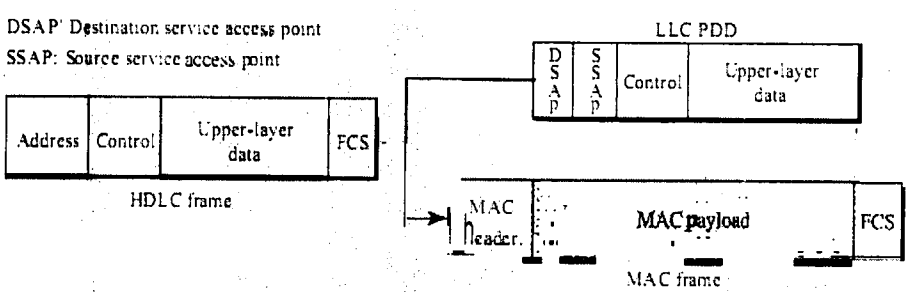


Figure 2.28 HDLC frame compared with LLC and MAC frames

Need for LLC The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services. For example, if a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols.

2.10 Wireless Networks

• Unguided media transport data without using a physical conductor. This type of communication is often referred to as wireless communication.

• It uses wireless electromagnetic signals to send data.

• There are three types of Unguided Media

(i) Radio waves

(ii) Micro waves

(iii) Infrared.

• Before understanding the different types of wireless transmission medium, let us first understand the ways in which wireless signals travel. These signals can be sent or propagated in the following three ways:

1. Ground-wave propagation

2. Sky-wave propagation

3. Line-of-sight propagation

1. Ground-wave propagation

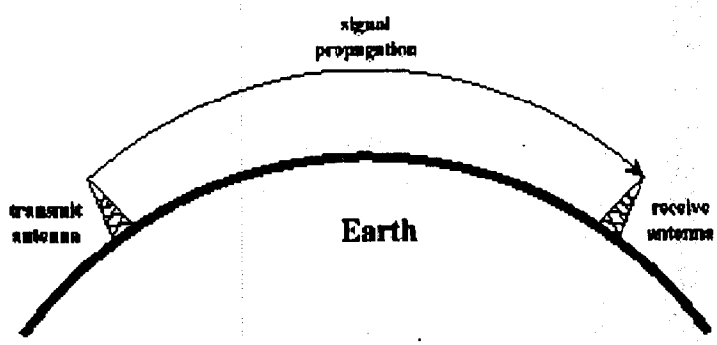


Figure 2.29 Ground Propagation of waves

Characteristics of Ground-wave propagation are as follows:

i. Follows contour of the earth

ii. Can propagate considerable distances

iii. Frequencies up to 2 MHz

iv. Example: AM radio

2. Sky-wave propagation

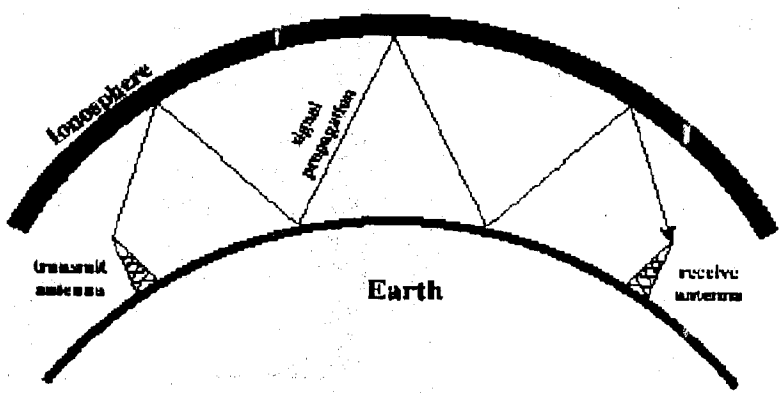


Figure 2.30 Sky-wave propagation of waves

Characteristics of Sky Propagation are as follows:

i. Signal reflected from ionized layer of atmosphere back down to earth

ii. Signal can travel a number of hops, back and forth between ionosphere and earth's surface

iii. Reflection effect caused by refraction

iv. Examples: Amateur radio, CB radio

3. Line-of-sight propagation

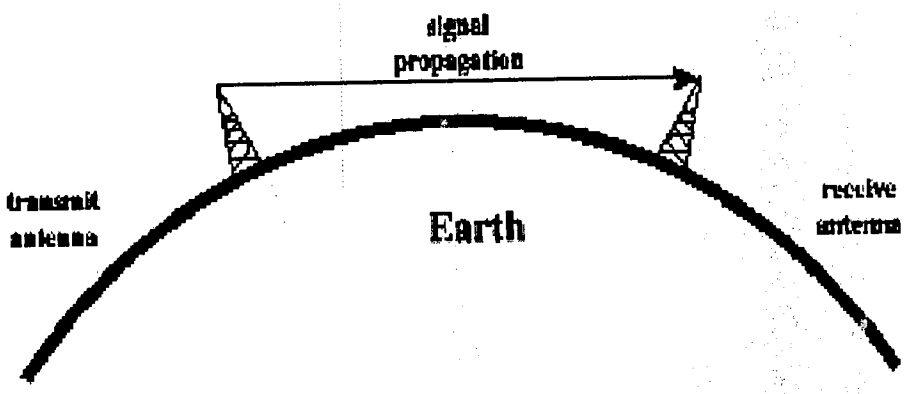


Figure 2.31 Line of Sight Propagation of waves

Characteristics of Line of Sight Propagation are as follows:

i. Transmitting and receiving antennas must be within line of sight

a. Satellite communication — signal above 30 MHz not reflected by ionosphere

b. Ground communication — antennas within effective line of site due to refraction

1. Radio waves:

• Electromagnetic wave ranging in frequencies between 3 KHz and 1GHz are normally called radio waves.

• Radio waves are Omni-directional when an antenna transmits radio waves they are propagated in all directions. This means that sending and receiving antenna do not have to he aligned. A sending antenna can send waves that can be received by any receiving antenna.

• Radio waves particularly those waves that propagate in sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio.

• Radio waves particularly those of low and medium frequencies can penetrate walls. It is an advantage because; an AM radio can receive signals inside a building. It is the disadvantage because we cannot isolate a communication to first inside or outside a building.

2. Microwaves:

• Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves.

• Microwaves are unidirectional; when an antenna transmits microwaves they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas.

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

• Parabolic dish antenna and horn antenna are used for this means of transmission

3. Infrared

• Infrared signals with frequencies ranges from 300 GHz to 400 GHz can be used for short range communication.

• Infrared signals, having high frequencies, cannot penetrate walls. This helps to prevent interference between one system and another. Infrared Transmission in one room cannot be affected by the infrared transmission in another room.

• Infrared band has an excellent potential for data transmission. Transfer digital data is possible with a high speed with a very high frequency. There are number of computer devices which are used to send the data through infrared medium e.g. keyboard mice, PCs and printers. There are some manufacturers provide a special part called the IrDA port that allows a wireless keyboard to communicate with a PC.

Summary

This unit introduced the network technology of a computer network, includes types of networking such as Local Area Network (LAN),

Metropolitan Area Network (MAN), Wide Area Network (WAN) and the IEEE standards for LAN and WAN. In addition to this, you will also study the various transmission media used in communication channels.

We can consider the data link layer as two sublayers. The upper sub layer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media. Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups: random access protocols, controlled access protocols, and channelization protocols. In random access or contention methods, no station is superior to another station and none is assigned the control over another.

ALOHA allows multiple access (MA) to the shared medium. There are potential collisions in this arrangement. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled. 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. Channelization is a multiple access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. We discussed three channelization protocols: FDMA, TDMA, and CDMA.

Review Questions

1. List three categories of multiple access protocols.

2. Define random access and list three protocols.

3. Explain Taking-Turns Protocols

4. How is the preamble field different from the SFD field?

5. What is the purpose of an NIC?

6 What is the difference between a unicast, multicast, and broadcast address?

7. What are the advantages of dividing an Ethernet LAN with a bridge?

8. What is the relationship between a switch and a bridge?

9. Why is there no need for CSMA/CD on a frill-duplex Ethernet LAN?

10. Compare the data rates for Standard Ethernet, Fast Ethernet, Gigabit Ethernet, and Ten-Gigabit Ethernet.

11. What are the common Standard Ethernet implementations?

12. What are the common Fast Ethernet implementations?

13. What are the common Gigabit Ethernet implementations?

14. What are the common Ten-Gigabit Ethernet implementations?

15. Assume that a SONET receiver resynchronizes its clock whenever a 1 bit appears; otherwise, the receiver samples the signal in the middle of what it believes is the bit's time slot.

(a) What relative accuracy of the sender's and receiver's clocks is required in order to receive correctly 48 0 bytes (one ATM AALS cell's worth) in a row?

(b) Consider a forwarding station A on a SONET STS-1 line, receiving frames from the downstream end B and retransmitting them upstream. What relative accuracy of A's and B's clocks is required to keep A from accumulating more than one extra frame per minute?

16. What are the difference between 802.5 and FDDI?

17. Explain wireless network transmission.

18. Describe Wide Area Networks.

Reference Books

1. Computer Networking: Schaum's outlines (TMH).

2. Kurose J F & Ross K.W: Computer Networking (Pearson)

3. Tanenbaum A S: Computer Networks (PHI) 4th Ed.

4. Data Communication & Networking – Behrouz Forouzan(TMH)

UNIT – III

Structure

3.1 Introduction

Up until the mid 1990s, three types of LAN interconnection devices were essentially available: hubs (and their cousins, repeaters), bridges and routers. More recently yet another interconnection device became widely available, namely, Ethernet switches. Ethernet switches, often trumpeted by network equipment manufacturers with great fanfare, are in essence high-performance multi-interface bridges. As do bridges, they forward and filter frames using LAN destination addresses, and they automatically build routing tables using the source addresses in the traversing frames. The most important difference between a bridge and switch is that bridges usually have a small number of interfaces, whereas switches may have dozens of interfaces. A large number interfaces generates a high aggregate forwarding rate through the switch fabric, therefore necessitating a high-performance design (especially for 100 Mbps and 1Gbps interfaces).

Switches can be purchased with various combinations of 10 Mbps, 100 Mbps and 1Gbps interfaces. For example, you can purchase switches with four 100 Mbps interfaces and twenty 10 Mbps interfaces; or switches with four 100 Mbps interfaces and one 1Gbps interface. Of course, the more the interfaces and the higher transmission rates of the various interfaces, the more you pay. Many switches also operate in a **full-duplex mode**; that is, they can send and receive frames at the same time over the same interface. With a full duplex switch (and corresponding full duplex Ethernet

adapters in the hosts), host A can send a file to host B while that host B simultaneously sends to host A.

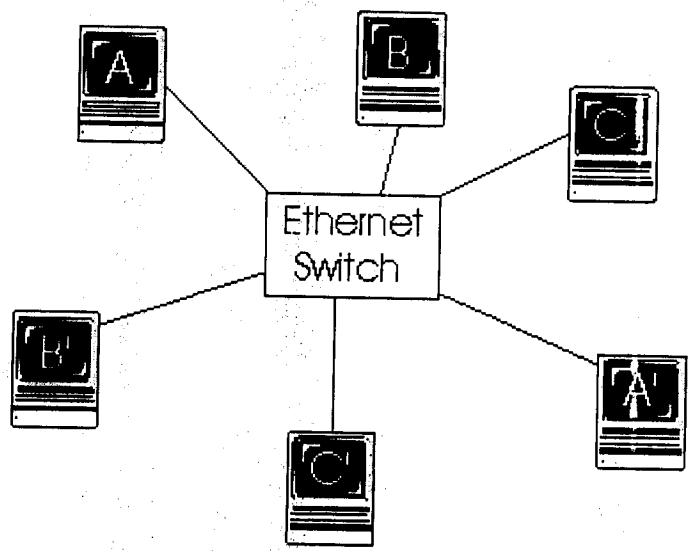


Figure 3.1 An Ethernet switch providing dedicated Ethernet access to six hosts.

One of the advantages of having a switch with a large number of interfaces is that it creates direct connections between hosts and the switch. When a host has a full-duplex direct connection to a switch, it can transmit (and receive) frames at the full transmission rate of its adapter; in particular, the host adapter always senses an idle channel and never experiences a collision. When a host has a direct connection to a switch (rather than a shared LAN connection), the host is said to have **dedicated access**. In Figure 3.1, an Ethernet switch provides dedicated access to six hosts. This dedicated access allows A to send a file to A' while that B is sending a file to B' and C is sending a file to C'. If each host has a 10Mbps adapter card, then the aggregate throughput during the three simultaneous file transfers is 30 Mbps. If A and A' have 100 Mbps adapters and the remaining hosts have 10 Mbps adapters, then the aggregate throughput during the three simultaneous file transfers is 120 Mbps.

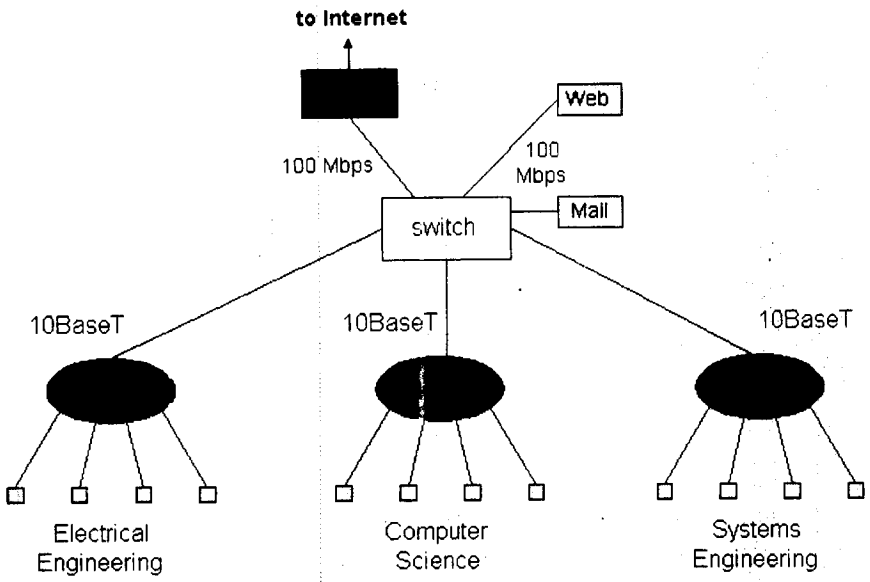


Figure 3.2 An institutional network using a combination of hubs, Ethernet switches and a router.

Figure 3.2 shows how an institution with several departments and several critical servers might deploy a combination of hubs, Ethernet switches and routers. In Figure 3.2, each of the three departments has its own 10 Mbps Ethernet segment with its own hub. Because each departmental hub has a connection to the switch, all intra-departmental traffic is confined to the Ethernet segment of the department (assuming the routing tables in the Ethernet switch are complete). The Web and mail servers each have dedicated 100 Mbps access to the switch. Finally, a router, leading to the Internet, has dedicated 100 Mbps access to the switch. Note that this switch has at least three 10 Mbps interfaces and three100 Mbps interfaces.

From the point of view of the average telephone engineer, the phone system is divided into two principal parts: outside plant (the local loops and trunks, since they are physically outside the switching offices) and inside plant (the switches), which are inside the switching offices. We have just looked at the outside plant. Now it is time to examine the inside plant.

Two different switching techniques are used nowadays: circuit switching and packet switching. We will give a brief introduction to each of them below. Then we will go into circuit switching in detail because that is how the telephone system works. We will study packet switching in detail in subsequent chapters.

3.2 Circuit Switching

When you or your computer places a telephone call, the switching equipment within the telephone system seeks out a physical path all the way from your telephone to the receiver's telephone. This technique is called circuit switching and is shown schematically in Figure 3.3. Each of the six rectangles represents a carrier switching office (end office, toll office, etc.). In this example, each office has three incoming lines and three outgoing lines. When a call passes through a switching office, a physical connection is (conceptually) established between the line on which the call came in and one of the output lines, as shown by the dotted lines.

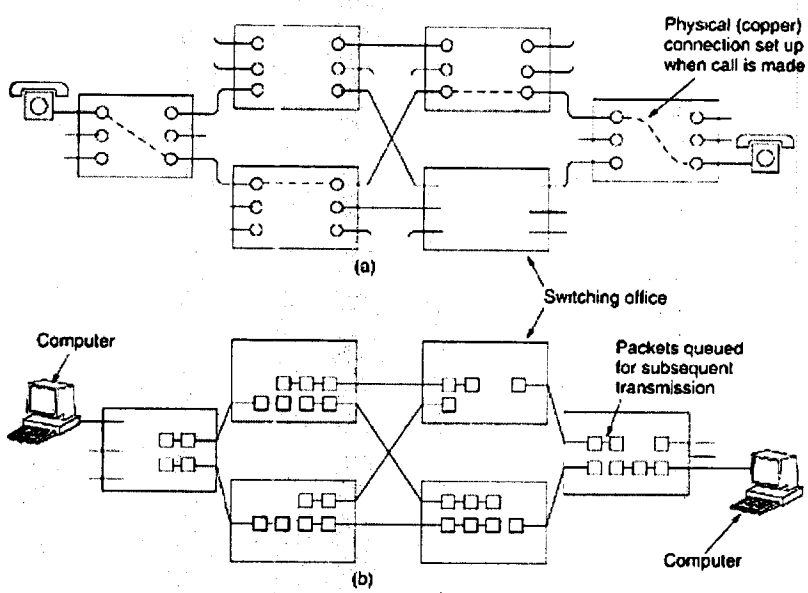


Figure 3.3 (a) Circuit switching. (b) Packet switching.

In the early days of the telephone, the connection was made by the operator plugging a jumper cable into the input and output sockets. In fact, a surprising little story is associated with the invention of automatic circuit switching equipment. It was invented by a 19th century Missouri undertaker named Almon B. Strowger. Shortly after the telephone was invented, when someone died, one of the survivors would call the town operator and say "Please connect me to an undertaker." Unfortunately for Mr. Strowger, there were two undertakers in his town, and the other one's wife was the town telephone operator. He quickly saw that either he was going to have to invent automatic telephone switching equipment or he was going to go out of business. He chose the first option. For nearly 100 years, the circuit-

switching equipment used worldwide was known as Strowger gear. (History does not record whether the now-unemployed switchboard operator got a job as an information operator, answering questions such as "What is the phone number of an undertaker?")

The model shown in Figure 3.4 (a) is highly simplified, of course, because parts of the physical path between the two telephones may, in fact, be microwave or fiber links onto which thousands of calls are multiplexed. Nevertheless, the basic idea is valid: once a call has been set up, a dedicated path between both ends exists and will continue to exist until the call is finished.

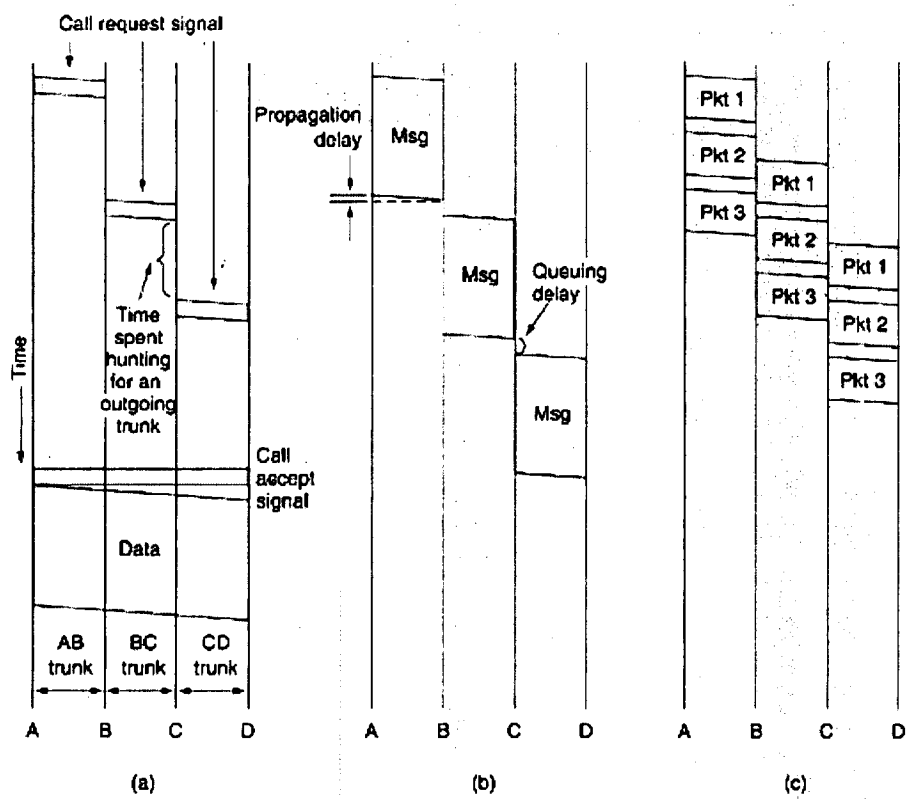


Figure 3.4 Timing of events in (a) circuit switching, (b) message switching, (c) packet switching.

The alternative to circuit switching is packet switching, shown in Figure 3.4(b). With this technology, individual packets are sent as need be, with no dedicated path being set up in advance. It is up to each packet to find its way to the destination on its own.

An important property of circuit switching is the need to set up an end-to-end path before any data can be sent. The elapsed time between the end of dialing and the start of ringing can easily be 10 sec, more on long-distance or international calls. During this time interval, the telephone system is hunting for a path, as shown in Figure 3.4(a). Note that before data transmission can even begin, the call request signal must propagate all the way to the destination and be acknowledged. For many computer applications (e.g., point-of-sale credit verification), long setup times are undesirable.

As a consequence of the reserved path between the calling parties, once the setup has been completed, the only delay for data is the propagation time for the electromagnetic signal, about 5msec per 1000 km. Also as a consequence of the established path, there is no danger of congestion that is, once the call has been put through, you never get busy signals. Of course, you might get one before the connection has been established due to lack of switching or trunk capacity.

3.3 Message Switching

An alternative switching strategy is message switching, illustrated in Figure 3.4(b). When this form of switching is used, no physical path is established in advance between sender and receiver. Instead, when the sender has a block of data to be sent, it is stored in the first switching office (i.e., router) and then forwarded later, one hop at a time. Each block is received in its entirety, inspected for errors, and then retransmitted. A network using this technique is called a store-and-forward network.

The first electromechanical telecommunication systems used message switching, namely, for telegrams. The message was punched on paper tape (off-line) at the sending office, and then read in and transmitted over a communication line to the next office along the way, where it was punched out on paper tape. An operator there tore the tape off and read it in on one of the many tape readers, one reader per outgoing trunk. Such a switching office was called a torn tape office. Paper tape is long gone and message switching is not used anymore, so we will not discuss it further in this book.

3.4 Packet Switching

With message switching, there is no limit at all on block size, which means that routers (in a modern system) must have disks to buffer long blocks. It

also means that a single block can tie up a router-router line for minutes, rendering message switching useless for interactive traffic. To get around these problems, packet switching was invented. Packet-switching networks place a tight upper limit on block size, allowing packets to be buffered in router main memory instead of on disk. By making sure that no user can monopolize any transmission line very long (milliseconds), packet-switching networks are well suited for handling interactive traffic. A further advantage of packet switching over message switching is shown in Figure 3.4(b) and (c): the first packet of a multipacket message can be forwarded before the second one has fully arrived, reducing delay and improving throughput. For these reasons, computer networks are usually packet switched, occasionally circuit switched, but never message switched.

Circuit switching and packet switching differ in many respects. To start with, circuit switching requires that a circuit be set up end to end before communication begins. Packet switching does not require any advance setup. The first packet can just be sent as soon as it is available.

The result of the connection setup with circuit switching is the reservation of bandwidth all the way from the sender to the receiver. All packets follow this path. Among other properties, having all packets follow the same path means that they cannot arrive out of order. With packet switching there is no path, so different packets can follow different paths, depending on network conditions at the time they are sent. They may arrive out of order.

Packet switching is more fault tolerant than circuit switching. In fact, that is why it was invented. If a switch goes down, all of the circuits using it are terminated and no more traffic can be sent on any of them. With packet switching, packets can be routed around dead switches.

Setting up a path in advance also opens up the possibility of reserving bandwidth in advance. If bandwidth is reserved, then when a packet arrives, it can be sent out immediately over the reserved bandwidth. With packet switching, no bandwidth is reserved, so packets may have to wait their turn to be forwarded.

Having bandwidth reserved in advance means that no congestion can occur when a packet shows up (unless more packets show up than expected). On the other hand, when an attempt is made to establish a circuit, the attempt can fail due to congestion. Thus, congestion can occur

at different times with circuit switching (at setup time) and packet switching (when packets are sent).

If a circuit has been reserved for a particular user and there is no traffic to send, the bandwidth of that circuit is wasted. It cannot be used for other traffic. Packet switching does not waste bandwidth and thus is more efficient from a system-wide perspective. Understanding this trade-off is crucial for comprehending the difference between circuit switching and packet switching. The trade-off is between guaranteed service and wasting resources versus not guaranteeing service and not wasting resources.

Packet switching uses store-and-forward transmission. A packet is accumulated in a router's memory, then sent on to the next router. With circuit switching, the bits just flow through the wire continuously. The store-and-forward technique adds delay.

Another difference is that circuit switching is completely transparent. The sender and receiver can use any bit rate, format, or framing method they want to. The carrier does not know or care. With packet switching, the carrier determines the basic parameters. A rough analogy is a road versus a railroad. In the former, the user determines the size, speed, and nature of the vehicle; in the latter, the carrier does. It is this transparency that allows voice, data, and fax to coexist within the phone system.

A final difference between circuit and packet switching is the charging algorithm. With circuit switching, charging has historically been based on distance and time. For mobile phones, distance usually does not play a role, except for international calls, and time plays only a minor role (e.g., a calling plan with 2000 free minutes costs more than one with 1000 free minutes and sometimes night or weekend calls are cheaper than normal). With packet switching, connect time is not an issue, but the volume of traffic sometimes is. For home users, ISPs usually charge a flat monthly rate because it is less work for them and their customers can understand this model easily, but backbone carriers charge regional networks based on the volume of their traffic. The differences are summarized in Figure 3.5.

Item	Circuit switched	Packet switched
Call setup	Required	Not needed
Dedicated physical path	Yes	No
Each packet follows the same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Bandwidth available	Fixed	Dynamic
Time of possible congestion	At setup time	On every packet
Potentially wasted bandwidth	Yes	No
Store-and-forward transmission	No	Yes
Transparency	Yes	No
Charging	Per minute	Per packet

Figure 3.5 A comparison of circuit-switched and packet-switched networks.

Both circuit switching and packet switching are important enough that we will come back to them shortly and describe the various technologies used in detail.

3.5 Switches Used in Circuit Switching

Circuit switching uses any of the three technologies

- Space-division switches
- Time-division switches
- a combination of both.

Cross bar Switch

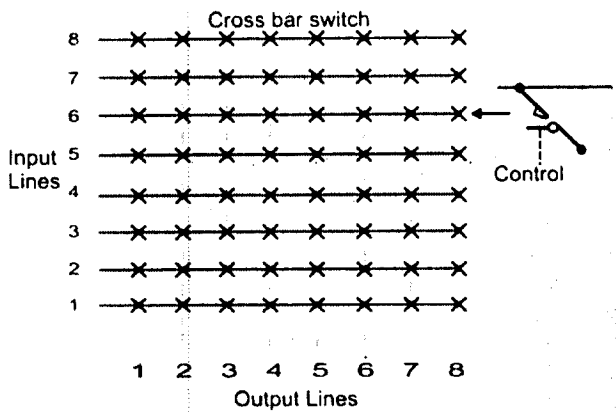


Figure 3.6 A Cross Bar Switch

- A crossbar switch is shown in Figure 3.6. Basic building block of the switch is a metallic cross point or semiconductor gate that can be enabled or disabled by a control unit.

- The data path is set by closing switches at intersecting intersection points.

Limitations

- The number of cross points grows with the square of the number of attached stations.
- Costly for a large switch.
- The failure of a cross point prevents connection between the two devices whose lines intersect at that cross point.
- The cross points are inefficiently utilized.
- Only a small fraction of cross points is engaged even if all of the attached devices are active.

Multi-Stage Space Division Switch

Some of the above problems can be overcome with the help of multistage space division switches. By splitting the crossbar switch into smaller units and interconnecting them, it is possible to build multistage switches with fewer cross points.

Figure 3.7 shows a three-stage space division switch.

- In this case, the number of cross points needed goes down from 64 to 40.
- There is more than one path through the network to connect two endpoints, thereby increasing reliability.

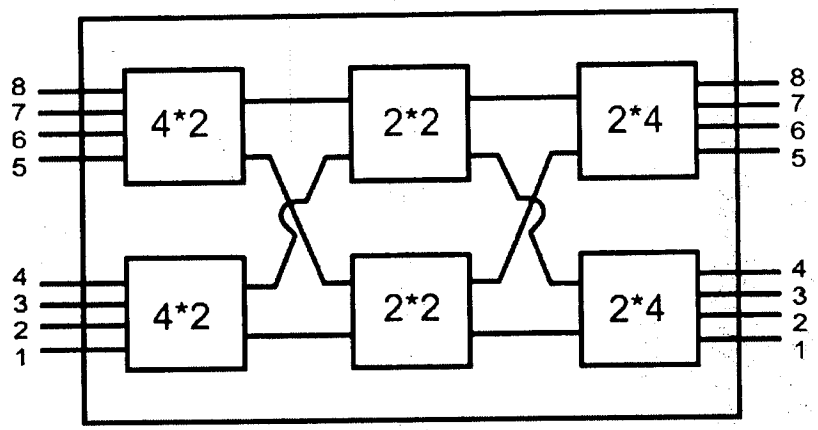


Figure 3.7 **A Three-Stage Space Division Switch**

- Multistage switches may lead to blocking. The problem may be tackled by increasing the number or size of the intermediate switches, which also increases the cost.
- The blocking feature is illustrated in Figure 3.8.

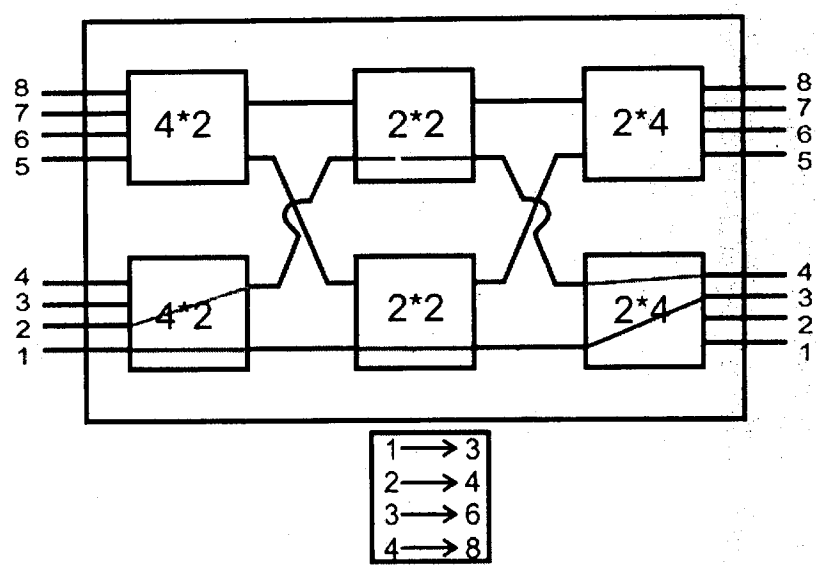


Figure 3.8 Block nature of the switch

- As shown in Figure 3.8, after setting up connections for 1-to-3 and 2-to-4, the switch cannot establish connections for 3-to-6 and 4-to-5.

Time-Division Switch

Time-division switching uses time-division multiplexing to achieve switching, i.e. different ongoing connections can use same switching path but at different interleaved time intervals.

- There are two popular methods of time-division switching namely
    - Time-Slot Interchange (TSI)
    - TDM bus

Time Slot Interchange

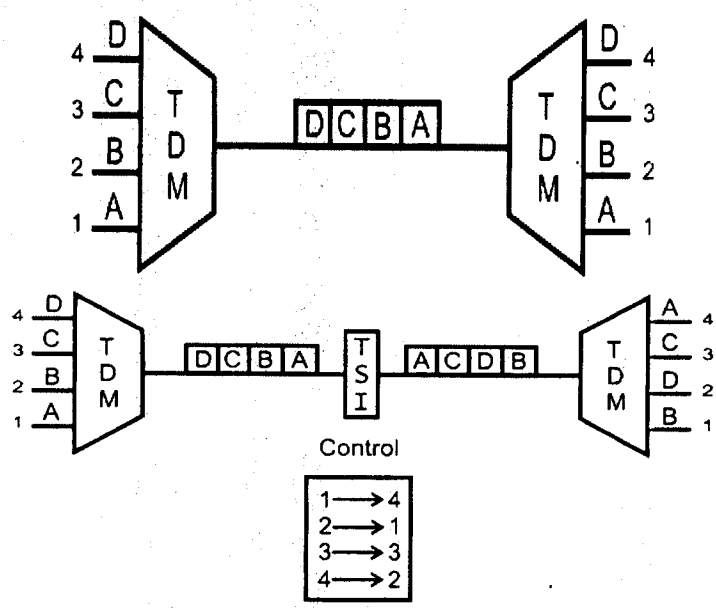


Figure 3.9 TDM Switching Using TSI

TSI changes the ordering of the slots based on desired connection and it has a random-access memory to store data and flip the time slots as shown in Figure 3.9. The operation of a TSI is depicted in Figure 3.10.

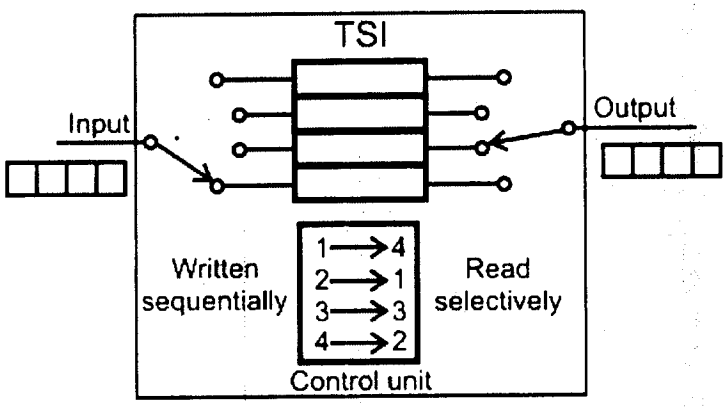


Figure 3.10 TSI Operation

As shown in the figure, writing can be performed in the memory sequentially, but data is read selectively.

TDM Bus

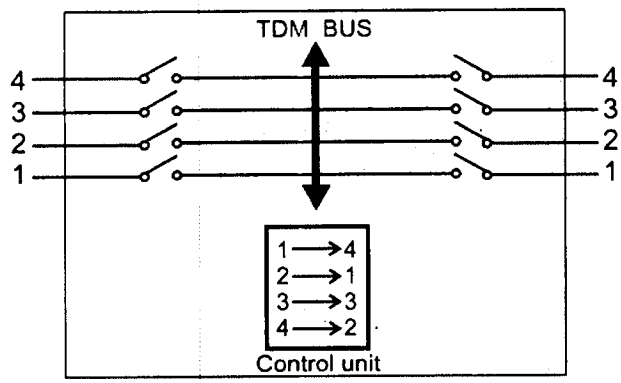


Figure 3.11 TDM Bus

In TDM bus, there are several input and outputs connected to a high-speed bus. During a time slot only one particular output switch is closed, so only one connection at a particular instant of time as shown in Figure 3.11.

3.6 ATM Switches

- An ATM cell switch has some number of input lines and some number of output lines (the both numbers are usually the same). ATM switches are generally synchronous in the sense of during a cycle, one cell is taken from each input line, passed into the internal

switching fabric, and eventually transmitted on the appropriate output line as shown in Figure 3.12.

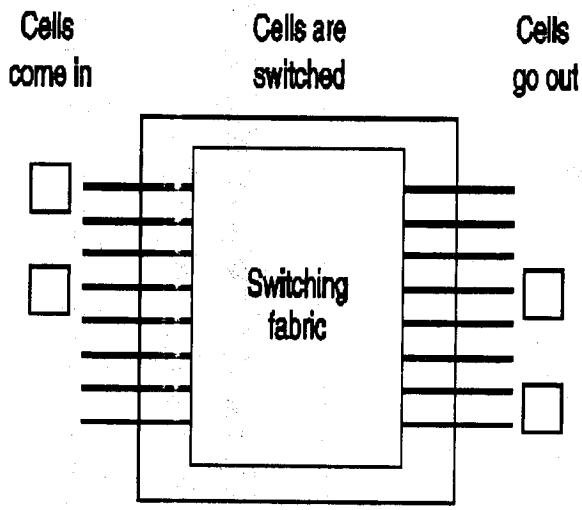


Figure 3.12 ATM Switch

Examples of ATM switches

- KNOCK-OUT SWITCH
- BANYAN SWITCH

Knockout Switch

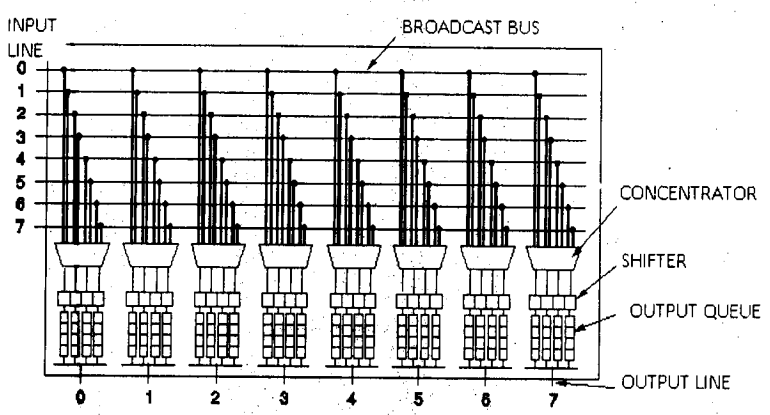


Figure 3.13 Knock-out Switch

The knockout switch is a fully connected architecture which attempts to combine the implementation simplicity of input queuing with the throughput

performance of output queuing. The knockout switch architecture achieves this goal by intentionally introducing a new source of packet loss, known as buffer blocking, in addition to packet loss mechanisms present in any switch architecture, namely buffer overflow and noise-induced random channel errors.

Each fixed-length cell arriving at one of the input ports is placed on a broadcast bus from which each of the output modules taps the cells intended for itself. It is obvious that multicast and broadcast cells are readily supported. The output module acts as statistical multiplexer, deferring cells that cannot be immediately placed onto the output link because of contention.

Each input to an output module receives the fixed-length cells broadcasted on the corresponding input bus. The job of each packet filter is simply to pass the cell to the concentrator if the cell is destined for that output, and to mark the cell as inactive otherwise. The role of the concentrator is to identify among its inputs those cells that are active and route them to its leftmost outputs, one cell per output line. Note that the concentrator has only L<N outputs. Should L+1 or more cells arrive simultaneously, only L of them will be processed via the concentrator; all others will be lost. This is the extra packet loss source in the knockout switch. By properly choosing L, the loss rate induced by the concentrator can be controlled and maintained at a reasonably low level. Furthermore, the value of L required to maintain a given loss rate is relatively small, independent of the number of inputs when the latter is large, and grows only logarithmically in the loss rate.

The concentrator inputs receive cells, which have already been passed by the packet filters and are known to be intended for the switch output port served by the concentrator. There are four (generally, L) stages in the concentrator shown in the diagram. Each stage is designed to operate like an elimination tournament. Specifically, each ß-element is programmed to set itself to the "bar" state if there is an active cell on its left input, and to "cross" otherwise. Whenever there is only one active cell at the inputs of a ß-element, it is allowed to pass downward. If both cells are active, the right-hand one is "knocked out" to the next stage and contends there. Each stage produces one "winner" among the active cells that enter it, and each subsequent stage receives one less active cell than the previous one. Therefore, when there are k active cells, they are guaranteed to come out on the outputs of the first k stages.

If the packet buffers in the output module diagram were to be loaded directly from the concentrator outputs, then the leftmost buffers would tend to fill up faster, and might even overflow despite the presence of empty buffer entries on the right. The shifter prevents that from happening by spreading each bulk of cells arriving at its input continuously to the right; in other words, if the last buffer to receive a packet happened to be m, then the next k cells arriving at the shifter's input will be directed to buffers m+1, ..., m+k (modulo L). Physically, the shifter can be implemented with an L×L Banyan network.

Because of the round-robin nature of the shifter and the fact that the buffers are filled cyclically, they can also be emptied cyclically. At each time slot, the output line fetches a cell from a buffer just right (cyclically) of the buffer last fetched from, beginning with buffer 1. Moreover, if the output circuitry encounters an empty buffer, the round-robin policy of buffer filling guarantees that all buffers are empty at that point, and the one just reached is precisely the next one to be filled again. The output pointer can then just stops there and wait for that buffer to receive a cell, after which the circular emptying of buffers can restart from that point.

PROBLEM

The Knock switch is similar to the Cross bar switch in terms of no of connections. As the nodes in the network grow in size, the number of connections increases and design become complex.

Banyan Switch

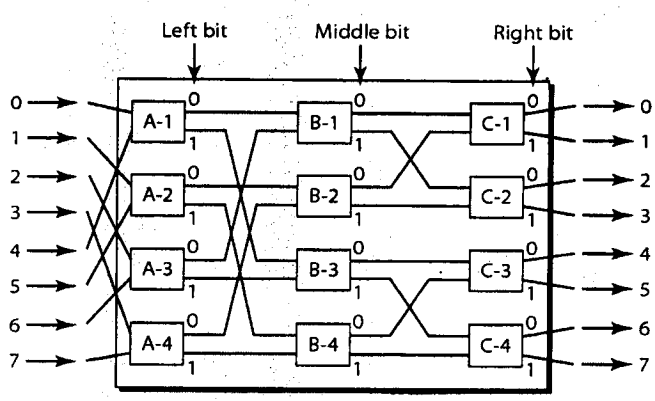


Figure 3.14 Banyan Switch

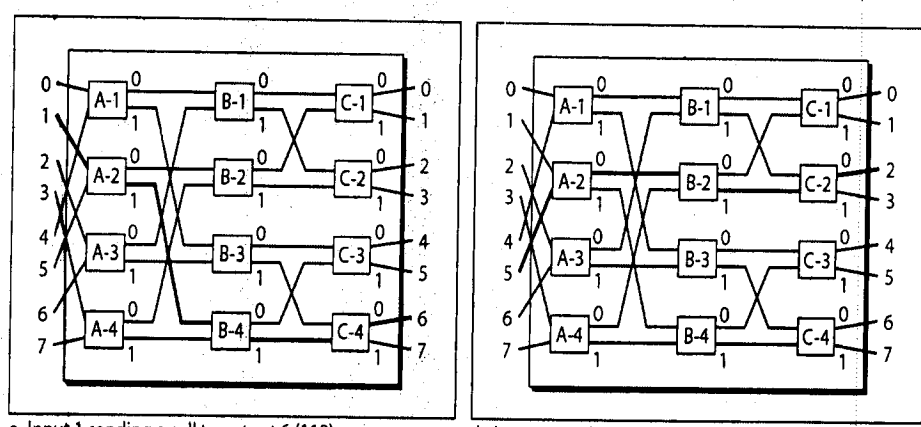
The banyan switch overcomes the problems involved switching. The Banyan switch uses clever arrangement of connections used to avoid collisions. We use a **perfect shuffle pattern** (Butterfly Network) for this purpose.

- A Banyan Switch with N inputs can have
  - N/2 Stages
  - $\log_2$ N switches per stage
- For example, an 8 node Banyan Switch can have 3 stages and 4 switches per stage as shown in Figure 3.14.
- Overall Complexity is n log n.

Routing in Banyan Switch Network

Consider a 8 X 8 banyan switch (Figure 3.15) which requires (8/2) $\log_2 8$ = 12 switching elements. It has three stages and 4 switches per stage. Inputs are applied as shown in Figure. Every packet that comes in has a header that contains one bit indicating what its destination is (either 0 or 1). If the switch reads the bit and it has value 0, it sends the packet to its higher output (which is 0 in this case), and to its lower output if the routing bit is one. By connecting these switching elements in series and parallel it is possible therefore, to route packets in more complicated ways depending on the desired routes to establish.

As shown Figure 3.15, each stage uses one bit. In this case, we have 8 inputs (0(000) to 7(111)). The first stage uses leftmost bit, second stage uses mid bit and third stage uses right most bit for routing.



a. Input 1 sending a cell to output 6 (110)

b. Input 5 sending a cell to output 2 (010)

Figure 3.15 Routing Examples

Figure 3.15shows routing of packet from Input 1 to output 6 and b shows Input 5 sending cell to output 2.

Virtual Circuit

- A virtual circuit, shown in Figure 3.16 is a seemingly always connected network which uses the concept of multiplexing.
- Two types of Virtual Circuit connections are offered
  - Permanent Virtual circuits
  - Switched Virtual Circuits

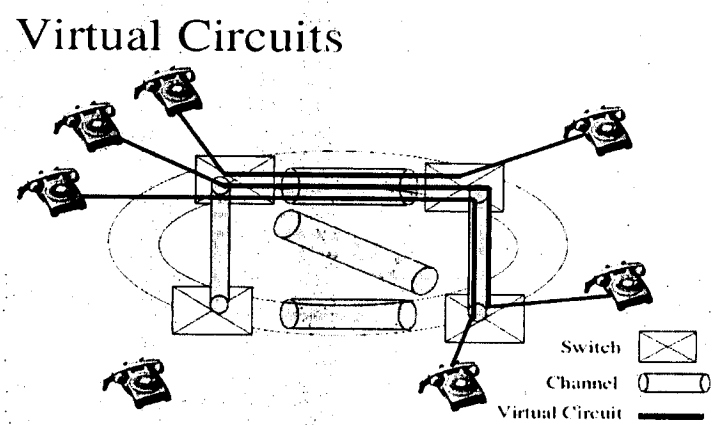


Figure 3.16 Virtual Circuit

Permanent VC:

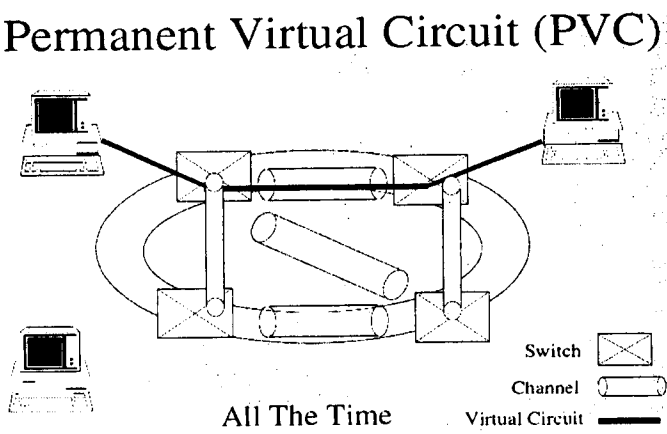
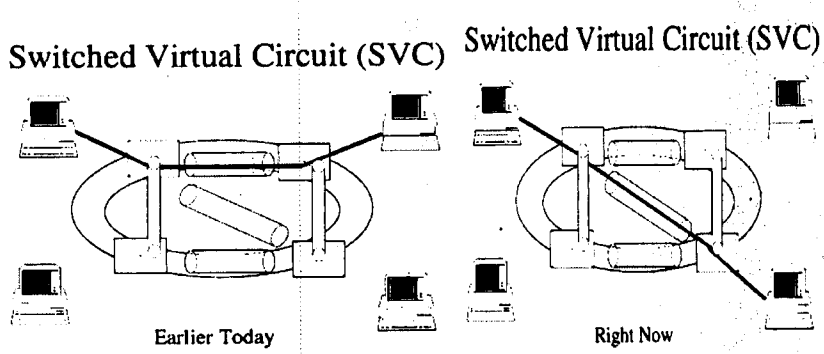


Figure 3.17 Permanent VC

Figure 3.17 shows a Permanent VC.

- End points and paths are fixed
- VC is available all the time
- Pre-Provisioned i.e., both bandwidth and services are guaranteed and fixed
- They are expensive
- Bandwidth may not be optimally used
- Analogy is leased line

Switched Virtual Circuits



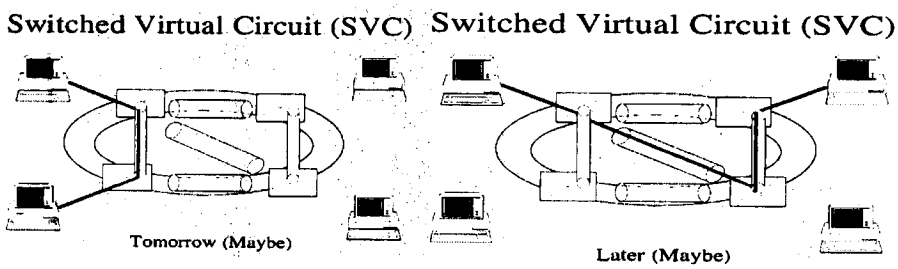


Figure 3.18 A Switched Virtual Circuit

As shown in Figure 3.18, the VCs in case Switched VC is not fixed. They may vary from time to time.

- Initially, requested bandwidth may not be guaranteed but a lower bandwidth may be negotiated and granted
- Connection Setup and tear down overhead
- Analogy is the regular and wireless phones

3.7 Bridges

In contrast to hubs, which are physical-level devices, bridges operate on Ethernet frames and thus are layer-2 devices. In fact, bridges are full-fledged packet switches that forward and filter frames using the LAN destination addresses. When a frame comes into a bridge interface, the bridge does not just copy the frame onto all of the other interfaces. Instead, the bridge examines the destination address of the frame and attempts to forward the frame on the interface that leads to the destination.

Figure 3.19 shows how the three academic departments of our previous example might be interconnected with a bridge. The three numbers next to the bridge are the interface numbers for the three bridge interfaces. When the departments are interconnected by a bridge, as in Figure 3.18, we again refer to the entire interconnected network as a LAN, and we again refer to each of the departmental portions of the network as LAN segments. But in contrast to the multi-tier hub design in Figure 3.18, each LAN segment is now an isolated collision domain.

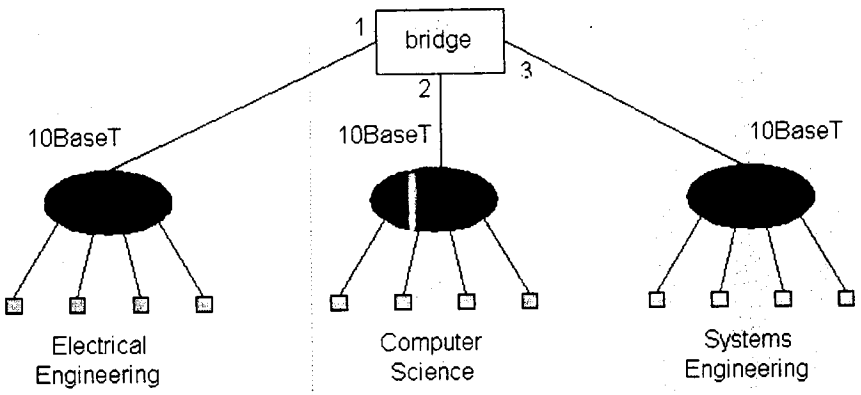


Figure 3.18 Three departmental LANs interconnected with a bridge.

Bridges can overcome many of the problems that plague hubs. First, bridges permit inter-departmental communication while preserving isolated collision domains for each of the departments. Second, bridges can interconnect different LAN technologies, including 10 Mbps and 100 Mbps Ethernets. Third, there is no limit to how big a LAN can be when bridges are used to interconnect LAN segments: in theory, using bridges, it is possible to build a LAN that spans the entire globe.

Bridge Forwarding and Filtering

Filtering is the ability to determine whether a frame should be forwarded to an interface or should just be dropped. When the frame should be forwarded, **forwarding** is the ability to determine which of the interfaces the frame should be directed to. Bridge filtering and forwarding are done with a **bridge table**. For each node on the LAN, the bridge table contains (1) the LAN address of the node, (2) the bridge interface that leads towards the node, (3) and the time at which the entry for the node was placed in the table. An example Table for the LAN in Figure 3.18 is shown in Figure 3.19. We note here that the addressees used by bridges are physical addresses (not network addresses). We will also see shortly that a bridge table is constructed in a very different manner than routing tables.

Address	Interface	Time
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

Figure 3.19 Portion of a bridge table for the LAN in Figure 3.18.

To understand how bridge filtering and forwarding works, suppose a frame with destination address DD-DD-DD-DD-DD-DD arrives to the bridge on

interface x. The bridge indexes its table with the LAN address DD-DD-DD-DD-DD-DD and finds the corresponding interface y.

- If x equals y, then the frame is coming from a LAN segment that contains adapter DD-DD-DD-DD-DD-DD. There being no need to forward the frame to any of the other interfaces, the bridge performs the filtering function by discarding the frame.
- If x does not equal y, then the frame needs to be routed to the LAN segment attached to interface y. The bridge performs its forwarding function by putting the frame in an output buffer that precedes interface y.

These simple rules allow a bridge to preserve separate collision domains for each of the different LAN segments connected to its interfaces. The rules also allow the nodes on different LAN segments to communicate.

Let's walk through these rules for the network in Figure 3.18 and its bridge table in Figure 3.19. Suppose that a frame with destination address 62-FE-F7-11-89-A3 arrives to the bridge from interface 1. The bridge examines its table and sees that the destination is on the LAN segment connected to interface 1 (i.e., the Electrical Engineering LAN). This means that the frame has already been broadcast on the LAN segment that contains the destination. The bridge therefore filters (i.e., discards) the frame. Now suppose a frame with the same destination address arrives from interface 2. The bridge again examines its table and sees that the destination is the direction of interface 1; it therefore forwards the frame to the output buffer preceding interface 1. It should be clear from this example that as long as the bridge table is complete and accurate, the bridge isolates the departmental collision domains while permitting the departments to communicate.

Recall that when a hub (or a repeater) forwards a frame onto a link, it just sends the bits onto the link without bothering to sense whether another transmission is currently taking place on the link. In contrast, when a bridge wants to forward a frame onto a link, it runs the CSMA/CD algorithm discussed in unit-III. In particular, the bridge refrains from transmitting if it senses that some other node on the LAN segment is transmitting; furthermore, the bridge uses exponential backoff when one of its transmissions results in a collision. Thus bridge interfaces behave very much like node adapters. But technically speaking, they are not node

adapters because neither a bridge nor its interfaces have LAN addresses. Recall that a node adapter always inserts its LAN address into the source address of every frame it transmits. This statement is true for router adapters as well as host adapters. A bridge, on the other hand, does not change the source address of the frame.

One significant feature of bridges is that they can be used to combine Ethernet segments using different Ethernet technologies. For example, if in Figure 3.18, Electrical Engineering has a 10Base2 Ethernet, Computer Science has a 100BaseT Ethernet, and Electrical Engineering has a 10BaseT Ethernet, then a bridge can be purchased that can interconnect the three LANs. With Gigabit Ethernet bridges, it is possible to have an additional 1Gbps connection to a router, which in turn connects to a larger university network. As we mentioned earlier, this feature of being able to interconnect different link rates is not available with hubs.

Also, when bridges are used as interconnection devices, there is no theoretical limit to the geographical reach of a LAN. In theory, we can build a LAN that spans the globe by interconnecting hubs in a long, linear topology, with each pair of neighboring hubs interconnected by a bridge. Because in this design each of the hubs has its own collision domain, there is no limit on how long the LAN can be. We shall see shortly, however, that it is undesirable to build very large networks exclusively using bridges as interconnection devices -- large networks need routers as well.

Self-Learning Bridge

A bridge has the very cool property of building its table automatically, dynamically and autonomously without any intervention from a network administrator or from a configuration protocol. In other words, bridges are **self-learning**. This is accomplished as follows.

- The bridge table is initially empty.
- When a frame arrives on one of the interfaces and the frame's destination address is not in the table, then the bridge forwards copies of the frame to the output buffers of all of the other interfaces. (At each of these other interfaces, the frame accesses the LAN segment using CSMA/CD.)
- For each frame received, the bridge stores in its table (1) the LAN address in the frame's source address field, (2) the interface from

which the frame arrived, (3) the current time. In this manner the bridge records in its table the LAN segment on which the sending node resides. If every node in the LAN eventually sends a frame, then every node will eventually get recorded in the table.

- When a frame arrives on one of the interfaces and the frame's destination address is in the table, then the bridge forwards the frame to the appropriate interface.

- The bridge deletes an address in the table if no frames are received with that address as the source address after a period of time (the aging time). In this manner, if a PC is replaced by another PC (with a different adapter), the LAN address of the original PC will eventually be purged from the bridge table.

Let's walk through the self-learning property for the network in Figure 3.18 and its corresponding bridge table in Figure 3.19. Suppose at time 9:39 a frame with source address 01-12-23-34-45-56 arrives from interface 2. Suppose that this address is not in the bridge table. Then the bridge appends a new entry in the table, as shown in Figure 3.20.

Address	**Interface**	**Time**
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
.....	.....	.....

Figure 3.20 Bridge learns about the location of adapter with address 01-12-23-34-45-56.

Continuing with this same example, suppose that the aging time for this bridge is 60 minutes and no frames with source address 62-FE-F7-11-89-A3 arrive to the bridge between 9:32 and 10:32. Then at time 10:32 the bridge removes this address from its table.

Bridges are **plug and play devices** because they require absolutely no intervention from a network administrator or user. When a network administrator wants to install a bridge, it does no more than connect the LAN segments to the bridge interfaces. The administrator does not have to configure the bridge tables at the time of installation or when a host is removed from one of the LAN segments. Because bridges are plug-and-play, they are also referred as **transparent bridges.**

Spanning Tree

One of the problems with a pure hierarchical design for interconnected LAN segments is that if a hub or a bridge near the top of the hierarchy fails, then much (if not all) of the interconnected LAN will go down. For this reason it is desirable to build networks with multiple paths between LAN segments. An example of such a network is shown in Figure 3.21.

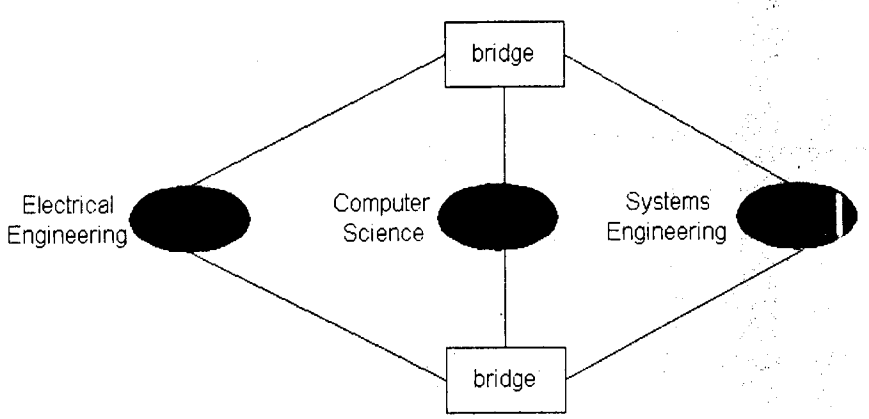


Figure 3.21 Interconnected LAN segments with redundant paths.

Multiple redundant paths between LAN segments (such as departmental LANs) can greatly improve fault tolerance. But, unfortunately, multiple paths have a serious side effect frames cycle and multiply within the interconnected LAN, thereby crashing the entire network. To see this, suppose that the bridge tables in Figure 3.21 are empty, and a host in Electrical Engineering sends a frame to a host in Computer Science. When the frame arrives to the Electrical Engineering hub, the hub will generate two copies of the frame and send one copy to each of the two bridges. When a bridge receives the frame, it will generate two copies; send one copy to the Computer Science hub and the other copy to the Systems Engineering hub. Since both bridges do this, there will be four identical frames in the LAN. This multiplying of copies will continue indefinitely since

the bridges do not know where the destination host resides. (To route the frame to the destination host in Computer Science, the destination host has to first generate a frame so that its address can be recorded in the bridge tables.) The number of copies of the original frame grows exponentially fast, crashing the entire network.

To prevent the cycling and multiplying of frames, bridges use a spanning tree protocol. In the **spanning tree protocol**, bridges communicate with each other over the LANs in order to determine a spanning tree, that is, a subset of the original topology that has no loops. Once the bridges determine a spanning tree, the bridges disconnect appropriate interfaces in order to create the spanning tree out of the original topology.

Bridges versus Routers

As we learned in unit-III, routers are store-and-forward packet switches that forward packets using IP addresses. Although a bridge is also a store-and-forward packet switch, it is fundamentally different from a router in that it forwards packets using LAN addresses. Whereas a router is layer-3 packet switch, a bridge is a layer-2 packet switch.

Even though bridges and routers are fundamentally different, network administrators must often choose between them when installing an interconnection device. For example, for the network in Figure 3.18, the network administrator could have just as easily used a router instead of a bridge. Indeed, a router would have also kept the three collision domains separate while permitting interdepartmental communication. Given that both bridges and routers are candidates for interconnection devices, what are the pros and cons of the two approaches?

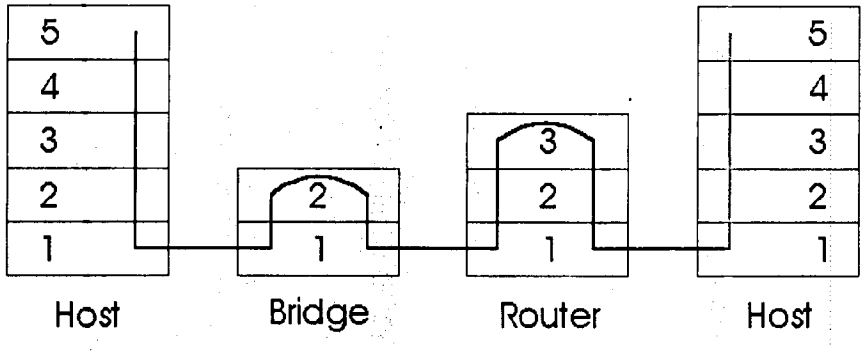


Figure 3.22 Packet processing and bridges, routers and hosts.

First consider the pros and cons of bridges. As mentioned above, bridges are plug-and-play, a property that is cherished by all the over-worked network administrators of the world. Bridges can also have relatively high packet filtering and forwarding rates as shown in Figure 3.22, bridges only have to process packets up through layer 2, whereas routers have to process frames up through layer 3. On the other hand, the spanning tree protocol restricts the effective topology of a bridged network to a spanning tree. This means that all frames most flow along the spanning tree, even when there are more direct (but disconnected) paths between source and destination. The spanning tree restriction also concentrates the traffic on the spanning tree links when it could have otherwise been spread through all the links of the original topology. Furthermore, bridges do not offer any protection against broadcast storms -- if one host goes haywire and transmits an endless stream of Ethernet broadcast packets, the bridges will forward all of the packets and the entire network will collapse.

Now consider the pros and cons of routers. Because IP addressing is hierarchical (and not flat as is LAN addressing), packets do not normally cycle through routers even when the network has redundant paths. (Actually, packets can cycle when router tables are misconfigured, but as we learned in unit-III, IP uses a special datagram header field to limit the cycling.) Thus, packets are not restricted to a spanning tree and can use the best path between source and destination. Because routers do not have the spanning tree restriction, routers have allowed the Internet to be built with a rich topology which includes, for example, multiple active links between Europe and North America. Another feature of routers is that they provide firewall protection against layer-2 broadcast storms. Perhaps the most significant drawback of routers is that they are not plug-and-play they and the hosts that connect to them need their IP addresses to be configured. Also, routers often have a larger prepackage processing time than bridges, because they have to process up through the layer-3 fields. Finally, there are two different ways to pronounce the word "router", either as "rootor" or as "rowter", and people waste a lot of time arguing over the proper pronunciation.

Given that both bridges and routers have their pros and cons, when should an institutional network (e.g., university campus network or a corporate campus network) use bridges, and when should it use bridges? Typically, small networks consisting of a few hundred hosts have a few LAN segments. Bridges suffice for these small networks, as they localize traffic

and increase aggregate throughput without requiring any configuration of IP addresses. But larger networks consisting of thousands of hosts typically include routers within the network (in addition to bridges). The routers provide a more robust isolation of traffic, control broadcast storms, and use more "intelligent" routes among the hosts in the network.

Connecting LAN Segments with Backbones

Consider once again the problem of interconnecting with bridges the Ethernets in the three departments in Figure 3.18. An alternative design is shown in Figure 3.23. This alternative design uses two two-interface bridges (i.e., bridges with two interfaces), with one bridge connecting Electrical Engineering to Computer Science, and the other bridge connecting Computer Science to Systems Engineering. Although two-interface bridges are very popular due to their low cost and simplicity, the design in Figure 3.23 is not recommended for two reasons. First, if the Computer Science hub were to fail, then Electrical Engineering and Systems Engineering would no longer be able to communicate. Second, and more important, all the inter-departmental traffic between Electrical and Systems Engineering has to pass through Computer Science, which may overly burden the Computer Science LAN segment.

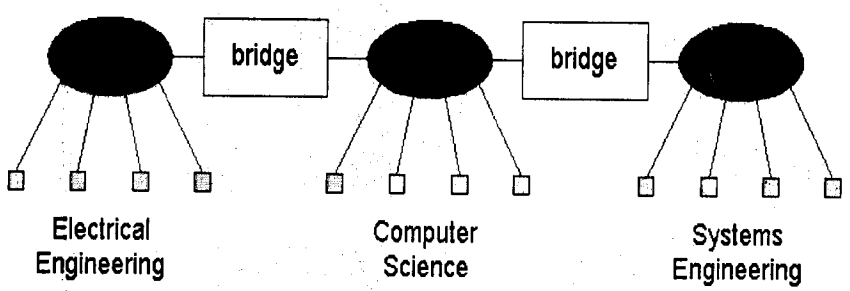


Figure 3.23 An example of an institutional LAN without a backbone.

One important principle when designing an interconnected LAN is that the various LAN segments should be interconnected with a backbone. A **backbone** is a network that has direct connections to all the LAN segments. When a LAN has a backbone, then each pair of LAN segments can communicate without passing through a third-party LAN segment. The design has shown Figure 3.18 uses a three-interface bridge for a backbone. In the homework problems at the end of this chapter we shall explore how to design backbone networks with two-interface bridges.

3.8 Naming and Addressing

Names and addresses both uniquely identify a host (or an interface on the host). Names are long and human understandable wastes space to carry them in packet headers hard to parse. Addresses are shorter and machine understandable if fixed size, easy to carry in headers and parse. Multiple names may point to same address can move a machine and just update the resolution table, for example:

```
%nslookup
    Default Server:  DUSK.CS.CORNELL.EDU
    Address:  128.84.227.13

    > underarm.com
    Name:     underarm.com
    Address:  206.128.187.146
```

Network addresses identify devices separately or as members of a group. Addressing is performed on various layers of the OSI model. Thus, schemes used for addressing vary on the basis of the protocol used and the OSI layer. On this basis, internetwork addresses can be categorized into three types. These are:

(a) Data link layer addresses

(b) Media Access Control (MAC) addresses

(c) Network layer addresses.

a) Data Link Layer Addresses

Data-link layer addresses sometimes are referred to as physical or hardware addresses, uniquely identify each physical network connection of a network device. Usually data-link addresses have a pre-established and fixed relationship to a specific device.

End systems generally have only one physical network connection and thus, have only one data-link address. Routers and other internetworking devices typically have multiple physical network connections and therefore, have multiple data-link addresses.

b) Media Access Control (MAC) Addresses

Media Access Control (MAC) addresses are used to identify network entities in LANs that implement the IEEE MAC addresses of the data link layer. These addresses are 48 bits in length and are expressed as 12 hexadecimal digits.

MAC addresses are unique for each LAN interface. These addresses consist of a subset of data link layer addresses. Figure 3.24 illustrates the relationship between MAC addresses, data-link addresses, and the IEEE sub-layers of the data link layer.

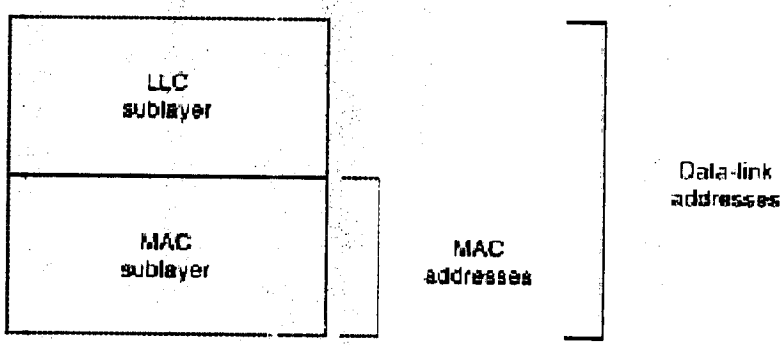


Figure 3.24 MAC addresses, data-link addresses, and the IEEE Sub-layers of the data link layer are all related

c) Network Layer Addresses

Network addresses are sometimes called virtual or logical addresses. These addresses are used to identify an entity at the network layer of the OSI model. Network addresses are usually hierarchical addresses.

Hierarchical vs. Flat Address

Usually Internetwork addresses are of two types:

(i) Hierarchical address

Hierarchical addresses are organized into a number of subgroups, each successively narrowing an address until it points to a single device as a house address.

(ii) Flat address

A flat address space is organized into a single group, such as, your enrolment no. Hierarchical addressing offers certain advantages over flat-addressing schemes. In hierarchical addressing, address sorting and recalling is simplified using the comparison operation. For example, "India" in a street address eliminates any other country as a possible location. Figure 3.25 illustrates the difference between hierarchical and flat address spaces.

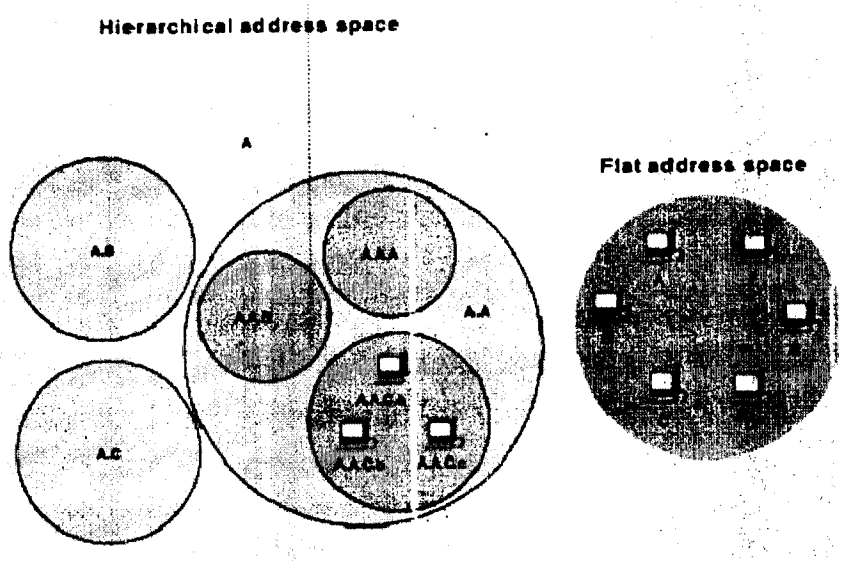


Figure 3.25 Hierarchical and flat address spaces differ in comparison operations

Static vs. Dynamic Address

In networking, the address to a device can be assigned in either of these two ways:

(I) Static address assignment: Static addresses are assigned by a network administrator according to a preconceived internetwork addressing plan. A static address does not change until the network administrator changes it manually.

(ii) Dynamic addresses: Dynamic addresses are obtained by devices when they are attached to a network, by means of some protocol-specific

process. A device using dynamic address often has a different address each time it connects to the network.

Address Classes

IP Address Construction: Dotted Decimal Notation

The Layer 3 address convention in the TCP/IP world uses a 32 bit binary number to logically identify each node on the network. The communicating nodes are referred to as **hosts** and the 32 bit binary address number is the **IP Address**. This 32 bit number is represented by breaking the 32 bits into four groups of eight bits each and representing each eight bit byte with the decimal value equivalent of the binary number.

This is referred to as the **Dotted-Decimal Notation** or as an Octet String. For example, assume a station is assigned the following address:

10000010000001000010110000000001

To represent this address in Dotted-Decimal Notation, the address is first broken up into four bytes, as follows:

10000010 00000100 00101100 00000001

Next, each byte is converted from binary to decimal, as follows:

10000010 = 130

00000100 = 4

00101100 = 44

00000001 = 1

The result is written in Dotted-Decimal Notation as:

130.4.44.1

Although the representation of the address consists of four decimal numbers separated by dots, the underlying meaning of the address can only be understood by evaluating the 32-bit binary number. This 32-bit number is used to identify each communicating device on the network.

Flat Networks versus Hierarchical Networks

In general there are two fundamental design relationships that can be identified in the construction of a network infrastructure. We can call these **flat** networks versus **hierarchical** networks. In a flat network every device is directly reachable by every other device. In a hierarchical network the world is divided into separate locations and devices are assigned to a specific location. The advantage of the hierarchical design is that the devices that interconnect the parts of the communications infrastructure need only know how to reach the intended destination location without having to keep track of the individual devices at each location. This device is a router. It makes a forwarding decision by looking at that part of the station address that identifies the location where the station resides.

All addressing in hierarchical networks may be considered to have two distinct parts. We might refer to these two parts as a **locator** portion and a **node** portion. The locator portion identifies the location at which the node resides. This "locator & node" address operates at Layer 3 of the OSI model, the Network Layer. Different vendors use different terms to refer to the locator and node portions of the address. For example, Novell NetWare calls them the "network" and "node" number. DECNet refers to the "area" and "node" number. TCP/IP uses two different terms to refer to the locator portion of the IP address. In a very fundamental sense these two terms are synonymous. They are the "network" and the "subnetwork". While it is true that the concept being expressed in IP routing is that of a "network" which is sub-divided into smaller locations referred to as "subnetworks" it must be remembered that a "subnetwork" may be further subdivided into smaller locations which would each also be referred to as a "subnetwork". The difference is mainly semantic in nature. Which of the following descriptions would you like to adopt?

1. The world is divided into separate networks, interconnected with routers.
2. A network is divided into subnetworks, interconnected with routers.
3. A subnetwork is divided into smaller subnetworks, interconnected with routers.

Whichever description you choose to represent the interconnection between locations the basic properties of connectivity remain identical. A

predefined location is connected to another predefined location using the decision making capabilities of a router. The terminology only helps to confuse the innocent (and that's us!).

In the IP world there is a configuration parameter that defines which bits in a Layer 3 address are used to differentiate between the locator portion and the node portion. This parameter is called the "address mask" (also called the "Subnet Mask"). We'll talk in-depth about the address mask under the SUBNET MASK topic (this is the Next Topic), but the simple explanation is that for each '1' bit in the mask, the corresponding bit in the address is considered to be part of the locator portion. Whether you refer to this as a subnet or a "Class A" or "Class B" network; it makes no difference. (The ADDRESS CLASSES topic describes network classes.) To understand IP addressing you must understand this fundamental rule, again:

For each '1' bit in the mask, the corresponding bit in the address is considered to be part of the locator portion of the address. The remaining bits are considered to identify a specific node at that location.

Assignment of Address Classes in the IP Version Four (IPV4) Specifications

Origins of Internet Protocol Version Four Addresses

The origins of the current implementation of the Internet Protocol (IP Version 4 or IPV4) and its associated classes of IP addressing can be traced to RFC 791: Internet Protocol (September 1981). As originally envisioned, these IP addresses were to be of fixed, 32-bit (4 Octets) length comprised of a Network Number and a Local Address or Host Number. The resulting range of addresses were then divided into three broad groupings or "Classes", each based upon the bit values within the first octet:

Class A - high order bit is "0", the remaining 7 bits are the network, and the last 24 bits are the host

Class B - high order two bits are "10", the remaining 14 bits are the network, and the last 16 bits are the host

Class C - high order three bits are "110", the remaining 21 bits are the network, and the last 8 bits are the host

*Note: There are two additional classes of IPV4 addressing, know as Class D & E that were specified in subsequent RFC's. These additional classes were intended for highly specialized functions and are identified as follows:

Class D - high order four bits are "1110", the remaining 20 bits identify the Multicast group
Class E - high order five bits are "11110", the remaining bits are reserved for experimental use

These two specialized classes of address are outside the scope of this article and will be addressed in a later Technical Compendium article.

The Roll of Masking in Determining Address Class

Implied within RFC 791, was the concept of "Masking", be used by Routers and Hosts. The masks were defined as follows:

- Class A mask = 255.0.0.0
- Class B mask = 255.255.0.0
- Class C mask = 255.255.255.0

*Note: These three original masks are often to as "Class A, B, C" or "Default" subnet masks. Details regarding the rolls of subnet masking are covered in the Technical Compendium articles "IP Address Construction: Dotted Decimal Notation, Subnet Masking, Creating Subnets, Special Subnet Masks and VLSM – Variable Length Subnet Masking".

These masks were applied by default based on the value of the leading bits in the IP address. If an address started with a binary 0, then stations assumed Class A masking. The starting bits 10 indicated Class B, and 110 indicated Class C. Consequently, the class of addressing masking being used could be determined by looking at the first octet in the address as shown below:

- Class A starts with 0 and ends with 0111 1111, hence the smallest value in the first octet is decimal 0 and the largest value is 127 yielding a potential range of 0-127.
- Class B starts with 10 and ends with 1011 1111, hence the smallest value in the first octet is decimal 128 and the largest value is 191 yielding a potential range of 128-191.

- Class C starts with 110 and ends with 1101 1111, hence the smallest value in the first octet is decimal 192 and the largest value is 223 yielding a potential range of 192-223.
- Class D (Reserved Multicast) starts with 1110 and ends with 1110 1111, hence the smallest value in the first octet is decimal 224 and the largest value is 239 yielding a potential range of 224-239.
- Class E (reserved Experimental) starts with 1111 and ends with 1111 1111, hence the smallest value in the first octet is decimal 240 and the largest value is 255 yielding a potential range of 240-255.

Some practical Examples of IP addressing -

1. A station with address 10.2.3.4 would be an example of a Class A address. The default mask for this station would be 255.0.0.0. The network identifier would be 10.0.0.0 (after applying the mask to the address) and the Host identifier would therefore be 2.3.4.
2. A station with address 140.6.15.3 would be an example of a Class B address. The default mask for this station would be 255.255.0.0. The network identifier would be 140.6.0.0 (after applying the mask to the address) and the Host identifier would therefore be 15.3.
3. A station with address 200.6.29.8 would be an example of a Class C address. The default mask for this station would be 255.255.255.0. The network identifier would be 200.6.29.0 (after applying the mask to the address) and the Host identifier would therefore be 8.

Determining The Number of Networks and Hosts in Each Class

Notice that this division of addressing into these three classes allows for the following potential number of addresses:

- **Class A**
  - 8 bits in the Network part, 24 bits in the Host part
  - 2^8 or 128 possible values in the Network part and 2^24 or 16777216 possible values in the Host part

- **Class B**
    - 16 bits in the Network part, 16 bits in the Host par
    - 2^{16} or 65536 possible values in the Network part and 2^16 or 65536 possible values in the Host part
- **Class C**
    - 24 bits in the Network part, 8 bits in the Host part
    - 2^{24} or 16777216 possible values in the network part and 256 values in the node part.

(Now, before you do the math on your own, let's work through the actual number of networks and hosts or nodes in each address class. First of all, realize that the number of values in a field is determined by raising the number2 to the exponential power determined by the number of bits in the field. Consequently we find:

2, raised to the 24th power: 224 = 16777216

2, raised to the 16th power: 216 = 65536

2, raised to the 8th power: 28 = 256)

*Note: The above values represent a maximum **theoretical** number of Network and Host addresses. AS we are about to examine, there are a number of factors that directly effect the true number of available addresses.

Factors Affecting Availible Address Values

As previously mentioned, there are a number of factors that have an in pact upon the available number of Network and Host addresses actually available within each address class:

1. **Overlapping Bit Values:** Foremost among these is the simple fact that bit values used for one class may NOT be used for the subsequent class(es):

a. Recall that for a Class A Network address, the Network identifier is 8 bits, however the first bit must always be zero, so that leaves only 7 bits to differentiate, so there are 2^7=128 possible class A Networks.

b. For a Class B Network address, the Network identifier is 16 bits, but the first two must be 10, so that leaves only 14 bits to differentiate, so there are 2^14=16384 possible Class B Networks.

c. For a Class C Network address, the Network Identifier is 24 bits, however, the first three must be 110, leaving only 21 bits to differentiate, so there are 2^21= 2,097,152 possible Class C Network addresses.

2. **Reserved Broadcast Numbers:** Additionally, there are two values in each set that will not be available. When the network or node portion of an IP address is set to all "1"s (ie, decimal 255 or hex FF) it indicates that this is a broadcast destination. There is also an older form of the broadcast address, from the UNIX environment, that used all "0"s. Although this form is considered obsolete it still may exist in some sites.

*Note: If you encounter a station that is still using zero's as a broadcast address you should reconfigure it (after you confirm that it really isn't being used by some alien system!).

Also, within the UNIX environment the configuration of a station includes specification of the correct broadcast address to use. A UNIX administrator might type "ifconfig broadcast 140.6.255.255" to specify the broadcast address for station 140.6.10.12. Other environments may allow only the use of the default values or may offer other ways of configuring the broadcast address.

Therefore, for each range of values, you must subtract 2 to arrive at the number of potential values that are available.

Summary of Actual Network and Host Values

All of this math can get more than a bit confusing, so the following chart summarizes the results of all of the mathematical manipulations and lists the available Network and Host addresses for each of the IPV4 Classes of Addressing:

Address Class	Range	Leading Bits	Implied Mask / Host Bits / Hosts
Class A	000 – 127	0000 0000 – 0111 1111	255.XXX.XXX.XXX
			0/127 Reserved, 126 Networks Possible
			16,711,680 Hosts Possible
Class B	128 – 191	1000 0000 – 1011 1111	255.255.XXX.XXX
			16,382 Networks Possible
			65,536 Hosts Possible
Class C	192 – 223	1100 0000 – 1101 1111	255.255.255.XXX
			2,097,150 Networks Possible
			0/255 reserved, 254 Hosts Possible
Class D	224 - 239	1110 0000 – 1110 1111	Reserved Network Multicast

Class E	240 – 247	1111 0000 – 1111 0111	Reserved For Experimental

Address Resolution Protocol(ARP)

In an Ethernet or Token Ring network, any station wishing to communicate with another station must know the DLC address of the Network Interface Card that will next receive the data frames. This presents a unique problem to higher layer protocol stacks because they must find some means of relating their Network Layer Addresses to the actual DLC address of the destination NIC. This essay discusses the purpose and functionality of TCP/IP's answer to this problem, Address Resolution Protocol (ARP).

Data Link Layer Addressing / Network Layer Addressing

Before we begin the discussion of ARP, an understanding of the difference between DLC (physical) layer addresses and Network Layer Addresses is required. We discussed, in the PREFACE to the Compendium, why we use the term "DLC" (Data Link Control) to refer to the Data Link Layer address; the network interface card address. An entity on a network generally has two addresses, the DLC (Data Link Control) address, refers to the actual burned-in address that is on the individual NIC in a station. DLC addresses are used in the lowest layer of communications and get the message from NIC to NIC. The second address, the Network Layer Address, is usually configured in the protocol stack software, and is used to convey the message from logical endpoint to logical endpoint. Some protocol stacks do not implement a Network Layer. Stations using these protocols address solely at the Data Link Layer.

Ethernet and Token-Ring cards only know how to recognize the DLC addresses therefore any communicator on a network must be able to specify the DLC address of both itself and the destination of its messages. Just knowing that you want to talk to IP address 128.200.10.4 is not sufficient.

Furthermore, a message traveling from one logical endpoint to another logical endpoint (station to station, for example) might in reality pass through several different NICs before reaching its destination. In the TCP/IP world, a packet destined for a station that is several gateways away would be addressed to the destination station at the Network Layer, but at the DLC layer would first be addressed to the first gateway, then the

second, then the third, etc... until the final gateway addressed the packet to the destination. Here, we see that at the Network Layer the packet is addressed to the logical endpoint of the conversation (the receiving station), but at the DLC layer, the packet had to pass through several NICs, so the DLC address changed.

The ARP Cache

In order to associate DLC addresses with Network Layer IP Addresses, a TCP/IP station maintains a record called an ARP Cache. The ARP Cache contains mappings of DLC addresses to IP addresses, and when a station wants to send a message to an IP address, it looks for the IP address in its ARP Cache, finds the associated DLC address, and can then properly address the message.

The Address Resolution Protocol comes into play when a station wants to communicate with a certain IP address, but does not have an entry for the IP address in its ARP Cache. A station in this position sends out an ARP frame addressed to the DLC broadcast, so that all IP stations on the network will receive the frame. The ARP broadcast basically says: "If you have the IP address I'm looking for, please respond and tell me your DLC address." If a station on the network has the IP address in the ARP frame, it responds directly to the ARPing station, which adds an entry in its ARP Cache, and then initiates a conversation. If no station responds, then the ARPing station times out.

Mapping Physical to Logical Address: RARP, BOOTP, and DHCP

There are occasions in which a host knows its physical address, but needs to know its logical address. This may happen in two cases:

1. A diskless station is just booted. The station can find its physical address by checking its interface, but it does not know its IP address.

2. An organization does not have enough IP addresses to assign to each station; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

RARP

Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address. Each host or router is

assigned one or more logical (IP) addresses, which are unique and independent of the physical (hardware) address of the machine. To create an IP datagram, a host or a router needs to know its own IP address or addresses. The IP address of a machine is usually read from its configuration file stored on a disk file.

However, a diskless machine is usually booted from RONI, which has minimum booting information. The ROM is installed by the manufacturer. It cannot include the IP address because the IP addresses on a network are assigned by the network administrator.

The machine can get its physical address (by reading its NIC, for example) which is unique locally. It can then use the physical address to get the logical address by using the RARP protocol. A RARP request is created and broadcast on the local network. Another machine on the local network that knows all the IP addresses will respond with a RARP reply. The requesting machine must be running a RARP client program; the responding machine must be running a RARP server program.

There is a serious problem with RARP: Broadcasting is done at the data link layer. The physical broadcast address, all is in the case of Ethernet, does not pass the boundaries of a network. This means that if an administrator has several networks or several subnets, it needs to assign a RARP server for each network or subnet. This is the reason that RARP is almost obsolete. Two protocols, BOOTP and DHCP, are replacing RARP.

BOOTP

The Bootstrap Protocol (BOOTP) is a client/server protocol designed to provide physical address to logical address mapping. BOOTP is an application layer protocol. The administrator may put the client and the server on the same network or on different networks. BOOTP messages are encapsulated in a UDP packet, and the UDP packet itself is encapsulated in an IP packet.

The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the servers IP address (the destination address). The client simply uses all as as the source address and a 111s as the destination address.

One of the advantages of BOOTP over RARP is that the client and server are application-layer processes. As in other application-layer processes, a

client can be in one network and the server in another, separated by several other networks. However, there is one problem that must be solved. The BOOTP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this case is called a relay agent. The relay agent knows the unicast address of a BOOTP server. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the BOOTP server. The packet, carrying a unicast destination address, is routed by any router and reaches the BOOTP server. The BOOT P server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the BOOTP client.

DHCP

BOOTP is not a dynamic configuration protocol. When a client requests its IP address, the BOOTP server consults a table that matches the physical dress of the client with its IP address. This implies that the binding between the physical address and the IP address of the client already exists. The binding is predetermined.

However, what if a host moves from one physical network to another? What if a host wants a temporary IP address? BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator. BOOTP is a static configuration protocol.

The Dynamic Host Configuration Protocol (DHCP) has been devised to provide static and dynamic address allocation that can be manual or automatic.

Static Address Allocation In this capacity DHCP acts as BOOTP does. It is backward- compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.

Dynamic Address Allocation DHCP has a second database with a pooi of available IP addresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

When a DHCP client sends a request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address ofthe client is returned. On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.

The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network (as is a subscriber to a service provider). DHCP provides temporary IP addresses for a limited time.

The addresses assigned from the pool are temporary addresses. The DHCP server issues a lease for a specific time. When the lease expires, the client must either stop using the IP address or renew the lease. The server has the option to agree or disagree with the renewal. If the server disagrees, the client stops using the address.

Manual and Automatic Configuration One major problem with the BOOTP protocol is that the table mapping the IP addresses to physical addresses needs to be manually configured. This means that every time there is a change in a physical or IP address, the administrator needs to manually enter the changes. DHCP, on the other hand, allows both manual and automatic configurations. Static addresses are created manually; dynamic addresses are created automatically.

Summary

Switching is necessary to connect two nodes or devices over the network that intends to communicate for a limited duration. Circuit switching is more suitable for voice communication, and can be done using space division, time division or a combination of both kinds of switches. Multistage switching is needed to optimize the number of cross points needed. Packet switching is used for data transmission and allows for prioritizing of data packets, alternate routing as needed and is also more efficient for the

bursty traffic pattern of data communication. Datagram's are self contained packets of data that are routed by the intermediate nodes of the network. Switched or permanent virtual circuits can also be utilized, where the route is established at the beginning of the session, but can be altered without disrupting the channel in case of failure of any part of the route.

Then we studied the concept of addressing. A network addresses identities devices separately or as members of a group. Internetwork addresses can be categorized into three types i.e., data link layer addresses, media access control (MAC) addresses and network layer addresses.

Review Questions

1. Why is switching necessary?
2. What is circuit switching?
3. What is packet switching? How does it differ from circuit switching?
4. What is meant by multistage switching? Is the capacity of such a switch limited?
5. List five important features of space division switching.
6. What are the characteristics of time division switching?
7. How can we combine space and time division switching? What are the advantages of such an approach?
8. What are three types of internetwork addresses? Explain in brief.
9. Differrentiate between following:

(i) Hierarchical address and Flat address

(ii) Static and Dynamic address.

10. What is the number of bits in an IPv4 address?
11. What are the differences between classful addressing and classless addressing in IPv4?
12. What is a mask in IPv4 addressing? What is a default mask in IPv4 addressing?

Reference Books

1. Computer Networking: Schaum's outlines (TMH).
2. Kurose J F & Ross K.W: Computer Networking (Pearson)
3. Tanenbaum A S: Computer Networks (PHI) 4th Ed.
4. Data Communication & Networking – Behrouz Forouzan(TMH)

UNIT – IV

Structure

4.1 Introduction

"Röting" is what fans do at a football game, what pigs do for truffles under oak trees in the Vaucluse, and what nurserymen intent on propagation do to cuttings from plants. "Rou' ting" is how one creates a beveled edge on a table top, or sends a corps of infantrymen into full-scale, disorganized retreat. Either pronunciation is correct for "routing," which refers to the process of discovering, selecting, and employing paths from one place to another (or to many others) in a network. The British prefer the spelling *routeing,* presumably to distinguish what happens in networks from what happened to the British in New Orleans in 1814. Since the *Oxford English Dictionary* is much heavier than any dictionary of American English, British English generally prevails in the documents produced by ISO and CCITT; wherefore, most of the international standards for routing protocols use the *routeing* spelling. Since this spelling would be unfamiliar to many readers.

A simple definition of routing is "learning how to get from here to there." In some cases, the term *routing* is used in a very strict sense to refer *only* to the process of obtaining and distributing information ("learning"), but not to the process of using that information to actually get from one place to another (for which a different term, *forwarding,* is reserved). Since it is difficult to grasp the usefulness of information that is acquired but never used, we employ the term *routing* to refer in general to all the things that are done to discover and advertise paths from here to there and to actually move packets from here to there when necessary. The distinction between routing and forwarding is preserved in the formal discussion of the functions performed by OSI end systems and intermediate systems, in which context the distinction is meaningful.

4.2 Routing Principles

In order to transfer packets from a sending host to the destination host, the network layer must determine the *path* or *route* that the packets are to follow. Whether the network layer provides a datagram service (in which case different packets between a given host-destination pair may take different routes) or a virtual circuit service (in which case all packets between a given source and destination will take the same path), the network layer must nonetheless determine the path for a packet. This is the job of the network layer **routing protocol.**

At the heart of any routing protocol is the algorithm (the "routing algorithm") that determines the path for a packet. The purpose of a routing algorithm is simple: given a set of routers, with links connecting the routers, a routing algorithm finds a "good" path from source to destination. Typically, a "good" path is one which has "least cost," but we will see that in practice, "real-world" concerns such as policy issues (e.g., a rule such as "router X, belonging to organization Y should not forward any packets originating from the network owned by organization Z") also come into play to complicate the conceptually simple and elegant algorithms whose theory underlies the practice of routing in today's networks.

The graph abstraction used to formulate routing algorithms is shown in Figure 4.1. (To view some graphs representing real network maps. Here, nodes in the graph represent routers - the points at which packet routing decisions are made - and the lines ("edges" in graph theory terminology) connecting these nodes represent the physical links between these routers. A link also has a value representing the "cost" of sending a packet across the link. The cost may reflect the level of congestion on that link (e.g., the current average delay for a packet across that link) or the physical distance traversed by that link (e.g., a transoceanic link might have a higher cost than a terrestrial link). For our current purposes, we will simply take the link costs as a given and won't worry about how they are determined.

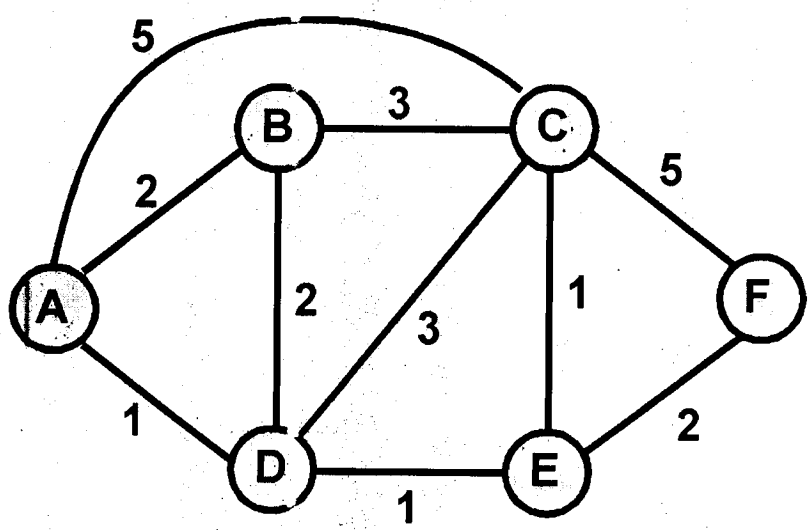


Figure 4.1: Abstract model of a network

Given the graph abstraction, the problem of finding the least cost path from a source to a destination requires identifying a series of links such that:

- the first link in the path is connected to the source
- the last link in the path is connected to the destination
- for all *i*, the *i* and *i-1*st link in the path are connected to the same node
- for the **least cost path**, the sum of the cost of the links on the path is the minimum over all possible paths between the source and destination. Note that if all link costs are the same, the least cost path is also the **shortest path** (i.e., the path crossing the smallest number of links between the source and the destination).

In Figure 4.1, for example, the least cost path between nodes A (source) and C (destination) is along the path ADEC. (We will find it notationally

easier to refer to the path in terms of the nodes on the path, rather than the links on the path).

4.3 Classification of Routing Algorithms

As a simple exercise, try finding the least cost path from nodes A to F, and reflect for a moment on how you calculated that path. If you are like most people, you found the path from A to F by examining Figure 4.1, tracing a few routes from A to F, and somehow convincing yourself that the path you had chosen was the least cost among all possible paths (Did you check all of the 12 possible paths between A and F? Probably not!). Such a calculation is an example of a centralized routing algorithm. Broadly, one way in which we can classify routing algorithms is according to whether they are centralized or decentralized:

- A **global routing algorithm** computes the least cost path between a source and destination using complete, global knowledge about the network. That is, the algorithm takes the connectivity between all nodes and all links costs as inputs. This then requires that the algorithm somehow obtain this information before actually performing the calculation. The calculation itself can be run at one site (a centralized global routing algorithm) or replicated at multiple sites. The key distinguishing feature here, however, is that a global algorithm has *complete* information about connectivity and link costs. In practice, algorithms with global state information are often referred to as **link state algorithms**, since the algorithm must be aware of the state (cost) of each link in the network.

- In a **decentralized routing algorithm,** the calculation of the least cost path is carried out in an iterative, distributed manner. No node has complete information about the costs of all network links. Instead, each node begins with only knowledge of the costs of its own directly attached links and then through an iterative process of calculation and exchange of information with its neighboring nodes (i.e., nodes which are at the "other end" of links to which it itself is attached) gradually calculates the least cost path to a destination, or set of destinations. We will study a decentralized routing algorithm known as a **distance vector algorithm.** It is called a distance vector algorithm because a node never actually knows a complete path from source to destination. Instead, it only knows the direction (which neighbor) to which it should forward a packet in order to reach a given destination along the least cost path, and the cost of that path from itself to the destination.

A second broad way to classify routing algorithms is according to whether they are **static** or **dynamic.** In static routing algorithms, routes change very slowly over time, often as a result of human intervention (e.g., a human manually editing a router's forwarding table). Dynamic routing algorithms change the routing paths as the network traffic loads (and the resulting delays experienced by traffic) or topology change. A dynamic algorithm can be run either periodically or in direct response to topology or

link cost changes. While dynamic algorithms are more responsive to network changes, they are also more susceptible to problems such as routing loops and oscillation in routes.

Only two types of routing algorithms are typically used in the Internet: a dynamic global link state algorithm, and a dynamic decentralized distance vector algorithm.

4.4 A Link State Routing Algorithm

Recall that in a link state algorithm, the network topology and all link costs are known, i.e., available as input to the link state algorithm. In practice this is accomplished by having each node **broadcast** the identities and costs of its attached links to *all* other routers in the network. This **link state broadcast** can be accomplished without the nodes having to initially know the identities of all other nodes in the network A node need only know the identities and costs to its directly-attached neighbors; it will then learn about the topology of the rest of the network by receiving link state broadcast from other nodes. (In Chapter 5, we will learn how a router learns the identities of its directly attached neighbors). The result of the nodes' link state broadcast is that all nodes have an identical and complete view of the network. Each node can then run the link state algorithm and compute the same set of least cost paths as every other node.

The link state algorithm we present below is known as Dijkstra's algorithm, named after its inventor (a closely related algorithm is Prim's algorithm; for a general discussion of graph algorithms). It computes the least cost path from one node (the source, which we will refer to as A) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the k^{th} iteration of the algorithm, the least cost paths are known to k destination nodes and among the least cost paths to all destination nodes, this k path will have the k smallest costs. Let us define the following notation:

- c(i,j): link cost from node i to node j. If nodes i and j are not directly connected, then c(i,j) = infty=α . We will assume for simplicity that c(i,j) equals c(j,i).
- D(v): the cost of path from the source node to destination v that has currently (as of this iteration of the algorithm) the least cost.
- p(v): previous node (neighbor of v) along current least cost path from source to v
- N: set of nodes whose shortest path from the source is definitively known

The link state algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes

NOTES

in the network. Upon termination, the algorithm will have calculated the shortest paths from the source node to every other node in the network.

Link State (LS) Algorithm:

```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infty

Loop
  find w not in N such that D(w) is a minimum
  add w to N
  update D(v) for all v adjacent to w and not in N:
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N
```

As an example, let us consider the network in Figure 4.1 and compute the shortest path from A to all possible destinations. A tabular summary of the algorithm's computation is shown in Table 4.1, where each line in the table gives the values of the algorithms variables at the end of the iteration. Let us consider the few first steps in detail:

step	N	D(B),p(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infty	infty
1	AD	2,A	4,D		2,D	infty
2	ADE	2,A	3,E			4,E
3	ADEB		3E			4E
4	ADEBC					4E
5	ADEBCF					

Table 4.1: Steps in running the link state algorithm on network in Figure 4.1

- **In the initialization step,** the currently known least path costs from A to its directly attached neighbors, B, C and D are initialized to 2, 5 and 1 respectively. Note in particular that the cost to C is set to 5 (even though we will soon see that a lesser cost path does indeed exist) since this is cost of the direct (one hop) link from A to C. The costs to E and F are set to infinity since they are not directly connected to A.

- **In the first iteration,** we look among those nodes not yet added to the set N and find that node with the least cost as of the end of the previous iteration. That node is D, with a cost of 1, and thus D is added to the set N. Line 12 of the LS algorithm is then performed to update D(v) for all nodes v, yielding the results shown in the second line (step 1) in Table 4.1. The cost of the path to B is unchanged. The cost of the path to C (which was 5 at the end of the initialization) through node D is found to have a cost of 4. Hence this lower cost path is selected and C's predecessor along the shortest path from A is set to D. Similarly, the cost to E (through D) is computed to be 2, and the table is updated accordingly.

- **In the second iteration**, nodes B and E are found to have the shortest path costs (2), and we break the tie arbitrarily and add E to the set N so that N now contains A, D, and E. The cost to the remaining nodes not yet in N, i.e., nodes B, C and F, are updated via line 12 of the LS algorithm , yielding the results shown in the third row in the above table.

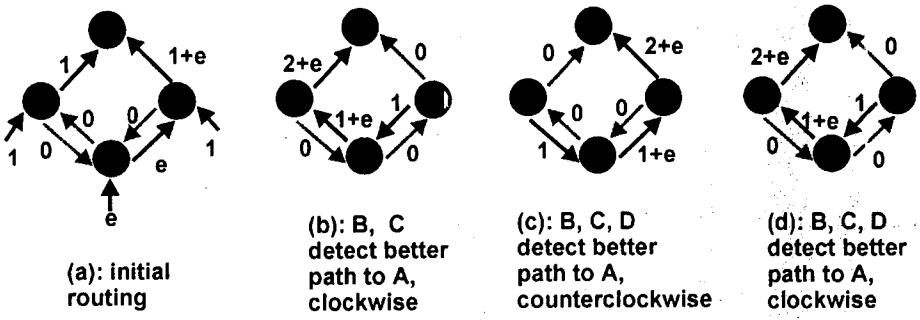
- and so on ...

When the LS algorithm terminates, we have for each node, its predecessor along the least cost path from the source node. For each predecessor, we also have *its* predecessor and so in this manner we can construct the entire path from the source to all destinations.

What is the computation complexity of this algorithm? That is, given *n* nodes (not counting the source), how much computation must be done in the worst case to find the least cost paths from the source to all destinations? In the first iteration, we need to search through all *n* nodes to determine the node, *w*, not in N that has the minimum cost. In the second iteration, we need to check *n-1* nodes to determine the minimum cost; in the third iteration *n-2* nodes and so on. Overall, the total number of nodes we need to search through over all the iterations is n*(n+1)/2, and thus we say that the above implementation of the link state algorithm has worst case complexity of order *n* squared: *O* (n^2). (A more sophisticated implementation of this algorithm, using a data structure known as a heap, can find the minimum in line 9 in logarithmic rather than linear time, thus reducing the complexity).

Before completing our discussion of the LS algorithm, let us consider a pathology that can arise with the use of link state routing. Figure 4.2 shows a simple network topology where link costs are equal to the load carried on

the link, e.g., reflecting the delay that would be experienced. In this example, link costs are not symmetric, i.e., c (A,B) equals c(B,A) only if the load carried on both directions on the AB link is the same. In this example, node D originates a unit of traffic destined for A, node B also originates a unit of traffic destined for A, and node C injects an amount of traffic equal to e, also destined for A. The initial routing is shown in Figure 4.2a, with the link costs corresponding to the amount of traffic carried.



(a): initial routing

(b): B, C detect better path to A, clockwise

(c): B, C, D detect better path to A, counterclockwise

(d): B, C, D detect better path to A, clockwise

Figure 4.2: Oscillations with Link State routing

When the LS algorithm is next run, node C determines (based on the link costs shown in Figure 4.2a) that the clockwise path to A has a cost of 1, while the counterclockwise path to A (which it had been using) has a cost of 1+e. Hence C's least cost path to A is now clockwise. Similarly, B determines that its new least cost path to A is also clockwise, resulting in the routing and resulting path costs shown in Figure 4.2b. When the LS algorithm is run next, nodes B, C and D all detect that a zero cost path to A in the counterclockwise direction and all route their traffic to the counterclockwise routes. The next time the LS algorithm is run, B, C, and D all then route their traffic to the clockwise routes.

What can be done to prevent such oscillations in the LS algorithm? One solution would be to mandate that link costs not depend on the amount of traffic carried -- an unacceptable solution since one goal of routing is to avoid highly congested (e.g., high delay) links. Another solution is to insure that all routers do not run the LS algorithm at the same time. This seems a more reasonable solution, since we would hope that even if routers run the LS algorithm with the same periodicity, the execution instants of the algorithm would not be the same at each node. Interestingly, researchers have recently noted that routers in the Internet can self-synchronize among themselves, i.e., even though they initially execute the algorithm with the same period but at different instants of time, the algorithm execution instants can eventually become, and remain, synchronized at the routers. One way to avoid such self-synchronization is to purposefully introduce randomization into the period between execution instants of the algorithm at each node.

Having now studied the link state algorithm, let's next consider the other major routing algorithm that is used in practice today - the distance vector routing algorithm.

4.5 A Distance Vector Routing Algorithm

While the LS algorithm is an algorithm using global information, the **distance vector (DV)** algorithm is iterative, *asynchronous, and distributed.* It is distributed in that each node receives some information from one or more of its *directly* attached neighbors, performs a calculation, and may then distribute the results of its calculation back to its neighbors. It is iterative in that this process continues on until no more information is exchanged between neighbors. (Interestingly, we will see that the algorithm is self terminating there is no "signal" that the computation should stop; it just stops). The algorithm is asynchronous in that it does not require all of the nodes to operate in lock step with each other. We'll see that an asynchronous, iterative, self terminating, distributed algorithm is much more "interesting" and "fun" than a centralized algorithm.

The principal data structure in the DV algorithm is the **distance table** maintained at each node. Each node's distance table has a row for each destination in the network and a column for each of its directly attached neighbors. Consider a node X that is interested in routing to destination Y via its directly attached neighbor Z. Node X's **distance table entry,** $D^x(Y,Z)$ is the sum of the cost of the direct one hop link between X and Z, c(X,Z), plus neighbor Z's currently known minimum cost path from itself (Z) to Y. That is:

$$D^x(Y,Z) = c(X,Z) + \min_w\{D^z(Y,w)\} \qquad (4.1)$$

The $\min_w$ term in equation 4.1 is taken over all of Z's directly attached neighbors (including X, as we shall soon see).

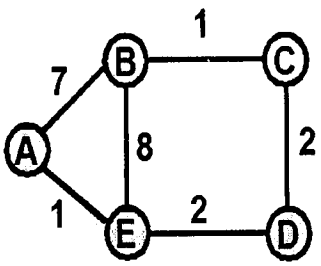
Equation 4.1 suggests the form of the neighbor-to-neighbor communication that will take place in the DV algorithm -- each node must know the cost of each of its neighbors minimum cost path to each destination Thus, whenever a node computes a new minimum cost to some destination, it must inform its neighbors of this new minimum cost.

Before presenting the DV algorithm, let's consider an example that will help clarify the meaning of entries in the distance table. Consider the network topology and the distance table shown for node E in Figure 4.3. This is the distance table in node E once the Dv algorithm has converged. Let's first look at the row for destination A.

- Clearly the cost to get to A from E via the direct connection to A has a cost of 1. Hence $D^E(A,A) = 1$.
- Let's now consider the value of $D^E(A,D)$ - the cost to get from E to A, given that the first step along the path is D. In this case, the

distance table entry is the cost to get from E to D (a cost of 2) plus whatever the minimum cost it is to get from D to A. Note that the minimum cost from D to A is 3 a path that passes right back through E! Nonetheless, we record the fact that the minimum cost from E to A given that the first step is via D has a cost of 5. We're left, though, with an uneasy feeling that the fact the path from E via D loops back through E may be the source of problems down the road (it will!).

- Similarly, we find that the distance table entry via neighbor B is $D^E(A,B) = 14$. Note that the cost is *not* 15. (Why?)



$D^E()$

destinat.	cost to destination via A	cost to destination via B	cost to destination via D
A	(1)	14	5
B	7	8	(5)
C	6	9	(4)
D	4	11	(2)

Figure 4.3: A distance table example

A circled entry in the distance table gives the cost of the least cost path to the corresponding destination (row). The column with the circled entry identifies the next node along the least cost path to the destination. Thus, a node's **routing table** (which indicates which outgoing link should be used to forward packets to a given destination) is easily constructed from the node's distance table.

In discussing the distance table entries for node E above, we informally took a global view, knowing the costs of all links in the network. The distance vector algorithm we will now present is *decentralized* and does not use such global information. Indeed, the only information a node will have are the costs of the links to its directly attached neighbors, and information it receives from these directly attached neighbors. The distance vector algorithm we will study is also known as the Bellman-Ford algorithm, after its inventors. It is used in many routing algorithms in practice, including: Internet BGP, ISO IDRP, Novell IPX, and the original ARPAnet.

Distance Vector (DV) Algorithm. At each node, X:

```
1                                        Initialization:
2       for all adjacent nodes v:
3    D^X(*,v) = infty      /* the * operator means "for all rows"
*/
4                  D^X(v,v)          =          c(X,v)
5          for all destinations, y
6    send min_w D(y,w) to each neighbor /* w over all X's
neighbors                                                  */
7
8                                                    loop
9    wait (until I see a link cost change to neighbor V
10       or until I receive update from neighbor V)
11
12         if (c(X,V) changes by d)
13    /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15    for all destinations y: D^X(y,V) = D^X(y,V) + d
16
17   else if (update received from V wrt destination Y)
18    /* shortest path from V to some Y has changed */
19    /* V has sent a new value for its min_w D^V(Y,w) */
20     /* call this received new value is "newval"   */
21    for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23   if we have a new min_w D^X(Y,w)for any destination Y
24     send new value of min_w D^X(Y,w) to all neighbors
25
26  forever
```

The key steps are lines 15 and 21, where a node updates its distance table entries in response to either a change of cost of an attached link or the receipt of an update message from a neighbor. The other key step is line 24, where a node sends an update to its neighbors if its minimum cost path to a destination has changed.

Figure 4.4 illustrates the operation of the DV algorithm for the simple three node network shown at the top of the figure. The operation of the algorithm is illustrated in a synchronous manner, where all nodes simultaneously receive messages from their neighbors, compute new distance table entries, and inform their neighbors of any changes in their new least path costs. After studying this example, you should convince yourself that the algorithm operates correctly in an asynchronous manner as well, with node computations and update generation/reception occurring at any times.

The circled distance table entries in Figure 4.4 show the current least path cost to a destination. An entry circled in red indicates that a new minimum cost has been computed (in either line 4 of the DV algorithm (initialization) or line 21). In such cases an update message will be sent (line 24 of the DV

algorithm) to the node's neighbors as represented by the red arrows between columns in Figure 4.4.

The leftmost column in Figure4.4 shows the distance table entries for nodes X, Y, and Z after the initialization step.

Let us now consider how node X computes the distance table shown in the middle column of Figure 4.4.4 after receiving updates from nodes Y and Z. As a result of receiving the updates from Y and Z, X computes in line 21 of the DV algorithm:

$$D^X(Y,Z) = c(X,Z) + \min_w D^Z(Y,w) = 7 + 1 = 8$$

$$D^{X(}Z,Y) = c(X,Y) + \min_w D^Y(Z,w) = 2 + 1 = 3$$

It is important to note that the only reason that X knows about the terms $\min_w D^Z(Y,w)$ and $\min_w D^Y(Z,w)$ is because nodes Z and Y have sent those values to X (and are received by X in line 10 of the DV algorithm). As an exercise, verify the distance tables computed by Y and Z in the middle column of Figure 4.4.

The value $D^{X(}Z,Y) = 3$ means that X's minimum cost to Z has changed from 7 to 3. Hence, X sends updates to Y and Z informing them of this new least cost to Z. Note that X need not update Y and Z about its cost to Y since this has not changed. Note also that Y's re-computation of its distance table in the middle column of Figure 4.4 *does* result in new distance entries, but *does not* result in a change of Y's least cost path to nodes X and Z. Hence Y does *not* send updates to X and Z.

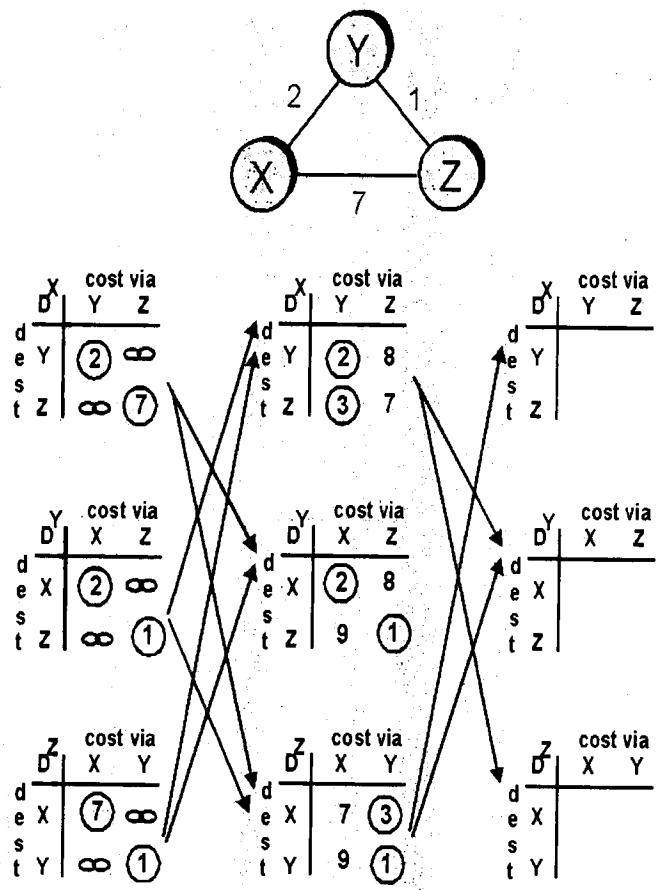


Figure 4.4: Distance Vector Algorithm: example

The process of receiving updated costs from neighbors, re-computation of distance table entries, and updating neighbors of changed costs of the least cost path to a destination continues until no update messages are sent. At this point, since no update messages are sent, no further distance table calculations will occur and the algorithm enters a quiescent state, i.e., all nodes are performing the wait in line 9 of the DV algorithm. The algorithm would remain in the quiescent state until a link cost changes, as discussed below.

4.6 Distance Vector Algorithm: Link Cost Changes and Link Failure

When a node running the DV algorithm detects a change in the link cost from itself to a neighbor (line 12) it updates its distance table (line 15) and, if there is a change in the cost of the least cost path, updates its neighbors (lines 23 and 24). Figure 4.5 illustrates this behavior for a scenario where the link cost from Y to X changes from 4 to 1. We focus here only on Y and Z's distance table entries to destination (row) X.

- At time t_0, Y detects the link cost change (the cost has changed from 4 to 1) and informs its neighbors of this change since the cost of a minimum cost path has changed.
- At time t_1, Z receives the update from Y and then updates its table. Since it computes a new least cost to X (it has decreased from a cost of 5 to a cost of 2), it informs its neighbors.
- At time t_2, Y has receives Z's update and has updates its distance table. Y's least costs have not changed (although its cost to X via Z has changed) and hence Y does *not* send any message to Z. The algorithm comes to a quiescent state.

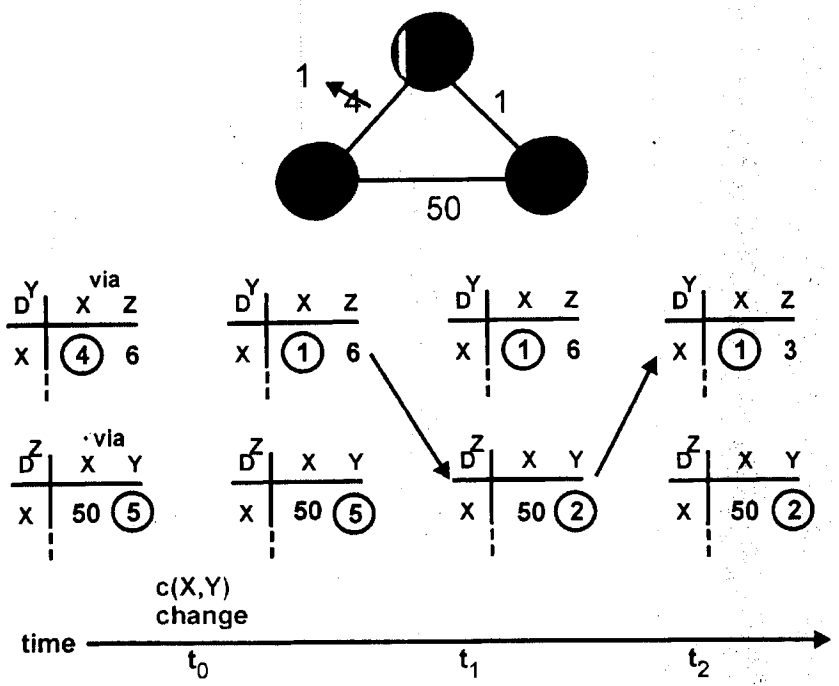


Figure 4.5: Link cost change: good news travels fast

In Figure 4.5, only two iterations are required for the DV algorithm to reach a quiescent state. The "good news" about the decreased cost between X and Y has propagated fast through the network.

Let's now consider what can happen when a link cost *increases*. Suppose that the link cost between X and Y increases from 4 to 60.

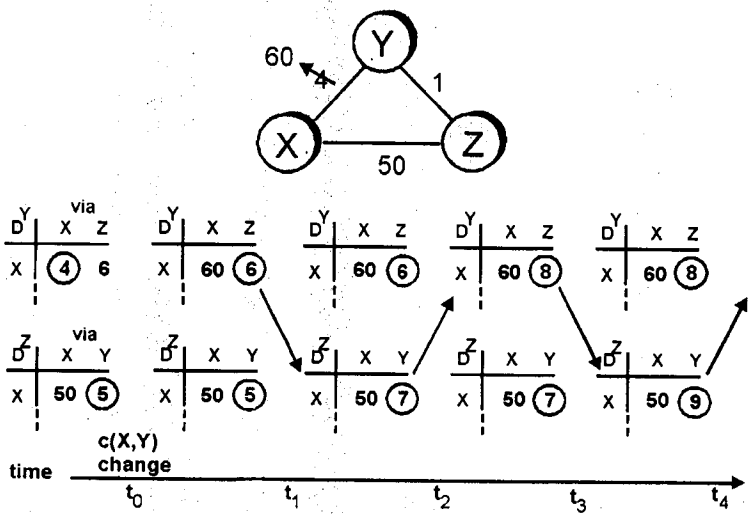


Figure 4.6: Link cost changes: bad news travels slow and causes loops

- At time t_0 Y detects the link cost change (the cost has changed from 4 to 60). Y computes its new minimum cost path to X to have a cost of 6 via node Z. Of course, with our global view of the network, we can see that this new cost via Z is *wrong*. But the only information node Y has is that its direct cost to X is 60 and that Z has last told Y that Z could get to X with a cost of 5. So in order to get to X, Y would now route through Z, fully expecting that Z will be able to get to X with a cost of 5. As of t_1 we have a **routing loop** in order to get to X, Y routes through Z, and Z routes through Y. A routing loop is like a black hole a packet arriving at Y or Z as of t_1 will bounce back and forth between these two nodes forever ..... Or until the routing tables are changed.

- Since node Y has computed a new minimum cost to X, it informs Z of this new cost at time t_1

- Sometime after t1, Z receives the new least cost to X via Y (Y has told Z that Y's new minimum cost is 6). Z knows it can get to Y with a cost of 1 and hence computes a new least cost to X (still via Y) of 7. Since Y's least cost to X has increased, it then informs Y of its new cost at t_2.

- In a similar manner, Y then updates its table and informs Z of a new cost of 9. Z then updates its table and informs Y of a new cost of 10, etc...

How long will the process continue? You should convince yourself that the loop will persist for 44 iterations (message exchanges between Y and Z) until Z eventually computes its path via Y to be larger than 50. At this point, Z will (finally!) determine that its least cost path to X is via its direct connection to X. Y will then route to X via Z. The result of the "bad news"

about the increase in link cost has indeed traveled slowly! What would have happened if the link cost change of c(Y,X) had been from 4 to 10,000 and the cost c(Z,X) had been 9,999? Because of such scenarios, the problem we have seen is sometimes referred to as the "count-to-infinity" problem.

Distance Vector Algorithm: Adding Poisoned Reverse.

The specific looping scenario illustrated in Figure 4.6 can be avoided using a technique known as poisoned reverse. The idea is simple if Z routes through Y to get to destination X, then Z will advertise to Y that its (Z's) distance to X is infinity. Z will continue telling this little "white lie" to Y as long as it routes to X via Y. Since Y believes that Z has no path to X, Y will never attempt to route to X via Z, as long as Z continues to route to X via Y (and lie about doing so).

Figure 4.7 illustrates how poisoned reverse solves the particular looping problem we encountered before in Figure 4.6. As a result of the poisoned reverse, Y's distance table indicates an infinite cost when routing to X via Z (the result of Z having informed Y that Z's cost to X was infinity). When the cost of the XY link changes from 4 to 60 at time t_0, Y updates its table and continues to route directly to X, albeit at a higher cost of 60, and informs Z of this change in cost. After receiving the update at t_1, Z immediately shifts it route to X to be via the direct ZX link at a cost of 50. Since this is a new least cost to X, and since the path no longer passes through Y, Z informs Y of this new least cost path to X at t_2. After receiving the update from Z, Y updates its distance table to route to X via Z at a least cost of 51. Also, since Z is now on Y's least path to X, Y poisons the reverse path from Z to X by informing Z at time t_3 that it (Y) has an infinite cost to get to X. The algorithm becomes quiescent after t_4, with distance table entries for destination X shown in the rightmost column in Figure 4.7.

Does poison reverse solve the general count-to-infinity problem? It does not. You should convince yourself that loops involving *three* or more nodes (rather than simply two immediately neighboring nodes, as we saw in Figure 4.7) will not be detected by the poison reverse technique.

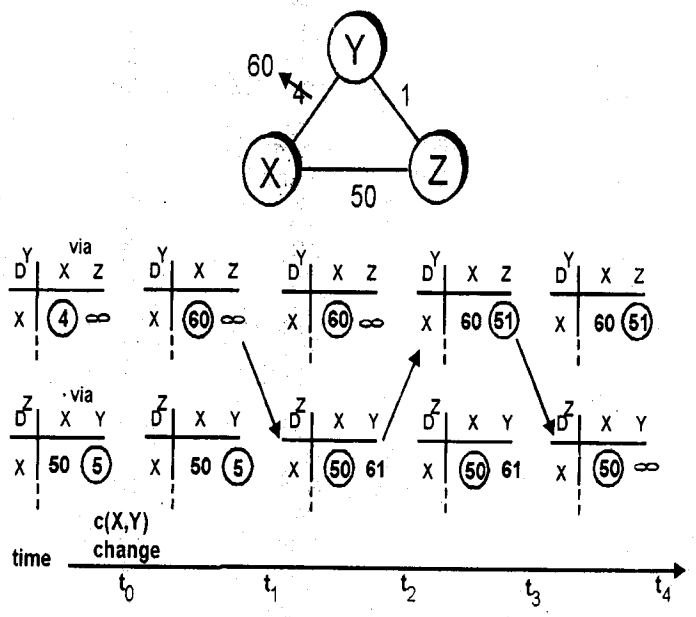


Figure 4.7: Poisoned reverse

4.7 A Comparison of Link State and Distance Vector Routing Algorithms

Let us conclude our study of link state and distance vector algorithms with a quick comparison of some of their attributes.

- **Message Complexity.** We have seen that LS requires each node to know the cost of each link in the network. This requires O(nE) messages to be sent, where n is the number of nodes in the network and E is the number of links. Also, whenever a link cost changes, the new link cost must be sent to *all* nodes. The DV algorithm requires message exchanges between directly connected neighbor at each iteration. We have seen that the time needed for the algorithm to converge can depend on many factors. When link costs change, the DV algorithm will propagate the results of the changed link cost *only* if the new link cost results in a changed least cost path for one of the nodes attached to that link.

- **Speed of Convergence.** We have seen that our implementation of the LS is an $O(n^2)$ algorithm requiring O(nE) messages, and potentially suffer from oscillations. The DV algorithm can converge slowly (depending on the relative path costs, as we saw in Figure 4.7) and can have routing loops while the algorithm is converging. DV also suffers from the count to infinity problem.

- **Robustness.** What can happen is a router fails, misbehaves, or is sabotaged? Under LS, a router could broadcast an incorrect cost for

one of its attached links (but no others). A node could also corrupt or drop any LS broadcast packets it receives as part of link state broadcast. But an LS node is only computing its own routing tables; other nodes are performing the similar calculations for themselves. This means route calculations are somewhat separated under LS, providing a degree of robustness. Under DV, a node can advertise incorrect least path costs to any/all destinations. (Indeed, in 1997 a malfunctioning router in a small ISP provided national backbone routers with erroneous routing tables. This caused other routers to flood the malfunctioning router with traffic, and caused large portions of the Internet to become disconnected for up to several hours.) More generally, we note that at each iteration, a node's calculation in DV is passed on to its neighbor and then indirectly to its neighbor's neighbor on the next iteration. In this sense, an incorrect node calculation can be diffused through the entire network under DV.

4.8 Other Routing Algorithms

The LS and DV algorithms we have studied are not only widely used in practice, they are essentially the only routing algorithms used in practice today.

Nonetheless, many routing algorithms have been proposed by researchers over the past 30 years, ranging from the extremely simple to the very sophisticated and complex. One of the simplest routing algorithms proposed is **hot potato routing.** The algorithm derives its name from its behavior -- a router tries to get rid of (forward) an outgoing packet as soon as it can. It does so by forwarding it on *any* outgoing link that is not congested, regardless of destination. Although initially proposed quite some time ago, interest in hot-potato-like routing has recently been revived for routing in highly structured networks, such as the so-called Manhattan street network.

Another broad class of routing algorithms is based on viewing packet traffic as flows between sources and destinations in a network. In this approach, the routing problem can be formulated mathematically as a constrained optimization problem known as a network flow problem. Let us define l_{ij} as the amount of traffic (e.g., in packets/sec) entering the network for the first time at node i and destined for node j. The set of flows, $\{l_{ij}\}$ for all i,j, is sometimes referred to as the network **traffic matrix.** In a network flow problem, traffic flows must be assigned to a set of network links subject to constraints such as:

- the sum of the flows between all source destination pairs passing though link m must be less than the capacity of link m;
- the amount of l_{ij} traffic entering any router r (either from other routers, or directly entering that router from an attached host) must equal the amount of l_{ij} traffic leaving router either via one of r's

outgoing links or to an attached host at that router. This is a **flow conservation** constraint.

Let us define l_{ij}^m as the amount of source i, destination j traffic passing through link m. The optimization problem then is to find the set of link flows, $\{l_{ij}^m\}$ for all links m and all sources, i, and designations, j, that satisfies the constraints above and optimizes a performance measure that is a function of $\{l_{ij}^m\}$. The solution to this optimization problem then defines the routing used in the network. For example, if the solution to the optimization problem is such that $l_{ij}^m = l_{ij}$ for some link m, then all i-to-j traffic will be routed over link m. In particular, if link m is attached to node i, and then m is the first hop on the optimal path from source i to destination j.

But what performance function should be optimized? There are many possible choices. If we make certain assumptions about the size of packets and the manner in which packets arrive at the various routers, we can use the so-called M/M/1 queuing theory formula to express the average delay at link as:

$$D_m = 1 / (R_m - S_i S_j l_{ij}^m),$$

where R_m is link m's capacity (measured in terms of the average number of packets/sec it can transmit) and $S_i S_j l_{ij}^m$ is the total arrival rate of packets (in packets/sec) that arrive to link m. The overall network wide performance measure to be optimized might then be the sum of all link delays in the network, or some other suitable performance metric. A number of elegant distributed algorithms exist for computing the optimum link flows (and hence routing determine the routing paths, as discussed above). The reader is referred to for a detailed study of these algorithms.

The final set of routing algorithms we mention here are those derived from the telephony world. These *circuit-switched* routing algorithms are of interest to packet-switched data networking in cases where per-link resources (e.g., buffers, or a fraction of the link bandwidth) are to reserved (i.e., set aside) for each connection that is routed over the link. While the formulation of the routing problem might appear quite different from the least cost routing formulation we have seen in this chapter, we will see that there are a number of similarities, at least as far as the path finding algorithm (routing algorithm) is concerned. Our goal here is to provide a brief introduction for this class of routing algorithms.

The circuit-switched routing problem formulation is illustrated in Figure 4.8. Each link has a certain amount of resources (e.g., bandwidth). The easiest (and a quite accurate) way to visualize this is to consider the link to be a bundle of circuits, with each call that is routed over the link requiring the dedicated use of one of the link's circuits. A link is thus characterized both by its total number of circuits, as well as the number of these circuits currently in use. In Figure 4.8, all links except AB and BD have 20 circuits; the number to the left of the number of circuits indicates the number of circuits currently in use.

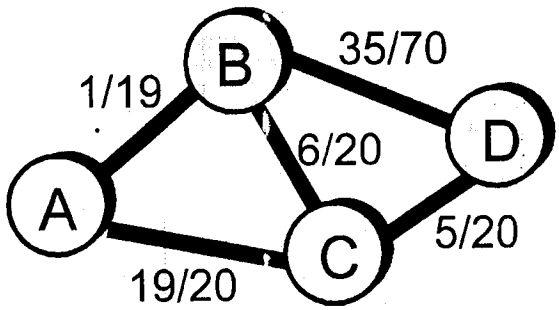


Figure 4.8: Circuit-switched routing

Suppose now that a call arrives at node A, destined to node D. What path should be taking? In **shortest path first** (SPF) routing, the shortest path (least number of links traversed) is taken. We have already seen how the Dijkstra LS algorithm can be used to find shortest path routes. In Figure 4.8, either that ABD or ACD path would thus be taken. In **least loaded path** (LLP) routing, the *load* at a link is defined as the ratio of the number of used circuits at the link and the total number of circuits at that link. The path load is the maximum of the loads of all links in the path. In LLP routing, the path taken is that with the smallest path load. In example 4.8, the LLP path is ABCD. In **maximum free circuit** (MFC) routing, the number of free circuits associated with a path is the minimum of the number of free circuits at each of the links on a path. In MFC routing, the path the maximum number of free circuits is taken. In Figure 4.8 the path ABD would be taken with MFC routing.

Given these examples from the circuit switching world, we see that the path selection algorithms have much the same flavor as LS routing. All nodes have complete information about the network's link states. Note however, that the potential consequences of old or inaccurate sate information are more severe with circuit-oriented routing a call may be routed along a path only to find that the circuits it had been expecting to be allocated are no longer available. In such a case, the call setup is blocked and another path must be attempted. Nonetheless, the main differences between connection-oriented, circuit-switched routing and connectionless packet-switched routing come not in the path selection mechanism, but rather in the actions that must be taken when a connection is set up, or torn down, from source to destination.

4.9 Hierarchical Routing

In the previous sections, we viewed "the network" simply as a collection of interconnected routers. One router was indistinguishable from another in the sense that all routers executed the same routing algorithm to compute routing paths through the entire network. In practice, this model and its view of a homogenous set of routers all executing the same routing algorithm is a bit simplistic for at least two important reasons:

Scale: As the number of routers becomes large, the overhead involved in computing, storing, and communicating the routing table information (e.g., link state updates or least cost path changes) becomes prohibitive. Today's public Internet consists of millions of interconnected routers and more than 50 million hosts. Storing routing table entries to each of these hosts and routers would clearly require enormous amounts of memory. The overhead required to broadcast link state updates among millions of routers would leave no bandwidth left for sending the data packets! A distance vector algorithm that iterated among millions of routers would surely never converge! Clearly, something must be done to reduce the complexity of route computation in networks as large as the public Internet.

Administrative autonomy: Although engineers tend to ignore issues such as a company's desire to run its routers as it pleases (e.g., to run whatever routing algorithm it chooses), or to "hide" aspects of the networks' internal organization from the outside, these are important considerations. Ideally, an organization should be able to run and administer its network as it wishes, while still being able to connect its network to other "outside" networks.

Both of these problems can be solved by aggregating routers into "regions" or "autonomous systems" (ASs). Routers within the same AS all run the same routing algorithm (e.g., a LS or DV algorithm) and have full information about each other exactly as was the case in our idealized model in the previous section. The routing algorithm running within an autonomous system is called an **intra-autonomous system routing protocol**. It will be necessary, of course, to connect ASs to each other, and thus one or more of the routers in an AS will have the added task for being responsible for routing packets to destinations outside the AS. Routers in an AS that have the responsibility of routing packets to destinations outside the AS are called **gateway routers**. In order for gateway routers to route packets from one AS to another (possibly passing through multiple other ASs before reaching the destination AS), the gateways must know how to route (i.e., determine routing paths) among themselves. The routing algorithm that gateways use to route among the various ASs is known as an **Inter-autonomous system routing protocol.**

In summary, the problems of scale and administrative authority are solved by defining autonomous systems. Within an AS, all routers run the same intra-autonomous system routing protocol. Special gateway routers in the various ASs run an inter-autonomous system routing protocol that determines routing paths among the ASs. The problem of scale is solved since an intra-AS router need only know about routers within its AS and the gateway router(s) in it's AS. The problem of administrative authority is solved since an organization can run whatever intra-AS routing protocol it chooses, as long as the AS's gateway(s) is able to run an inter-AS routing protocol that can connect the As to other ASs..

Figure 4.9 illustrates this scenario. Here, there are three routing ASs, A, B and C. Autonomous system A has four routers, A.a, A.b, A.c and A.d, which run the intra-AS routing protocol used within autonomous system A.

These four routers have complete information about routing paths within autonomous system A. Similarly, autonomous systems B and C have three and two routers, respectively. Note that the intra-AS routing protocols running in A, B and C need not be the same. The gateway routers are A.a, A.c, B.a and C.b. In addition to running the intra-AS routing protocol in conjunction with other routers in their ASs, these four routers run an inter-AS routing protocol among themselves. The topological view they use for inter-AS routing protocol is shown at the higher level, with "links" shown in light gray. Note that a "link" at the higher layer may be an actual physical link, e.g., the link connection A.c and B.a, or a logical link, such as the link connecting A.c and A.a.

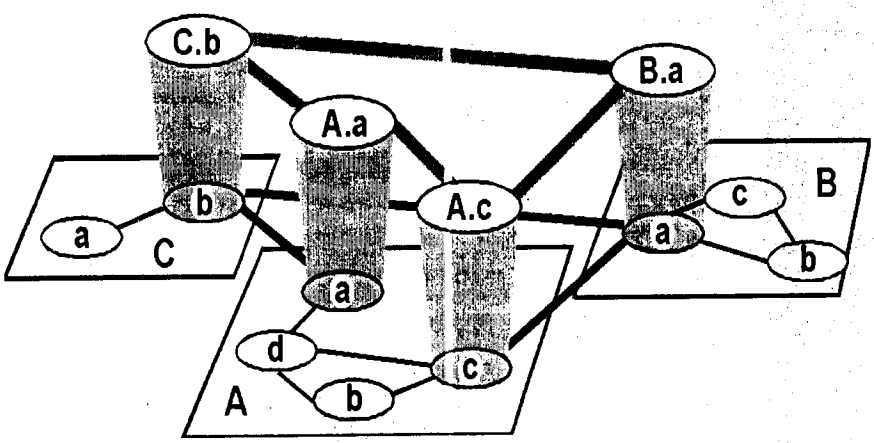


Figure 4.9: Intra-AS and Inter-AS routing.

Figure 4.10 illustrates that the gateway router A.c must run an intra-AS routing protocol with its neighbors A.b and A.d, as well as an inter-AS protocol with gateway router B.a.

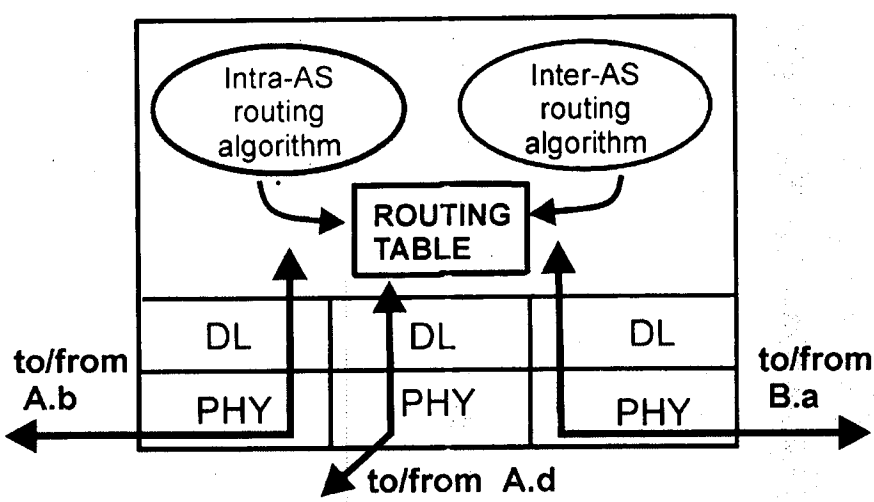


Figure 4.10: Internal architecture of gateway router A.c

Suppose now that a host h1 attached to router A.d needs to route a packet to destination h2 in autonomous system B, as shown in Figure 4.11. Assuming that A.d's routing table indicates that router A.c is responsible for routing its (A.d's) packets outside the AS, the packet is first routed from A.d to A.c using A's intra-AS routing protocol. It is important to note that router A.d does not know about the internal structure of autonomous systems B and C and indeed need not even know about the topology connecting autonomous systems A, B and C. Router A.c will receive the packet and see that it is destined to an autonomous system outside of A. A's routing table for the intra-AS protocol would indicate that a packet destined to autonomous system B should be routed along the A.c to B.a link. When the packet arrives at B.a, B.a's *inter-AS* routing sees that the packet is destined for autonomous system B. The packet is then "handed over" to the *intra-AS* routing protocol within B, which routes the packet to its final destination, h2.

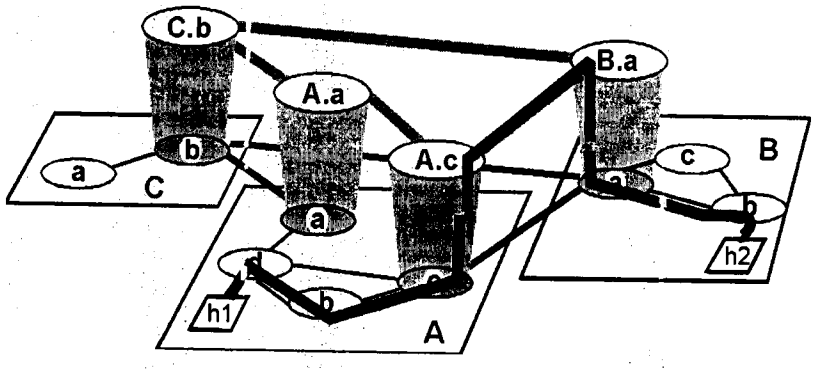


Figure 4.11: The route from A.d to B.b: intra-AS and inter-AS path segments.

4.10 Multicast Routing

Some applications require that widely-separated processes work together in groups, for example, a group of processes implementing a distributed database system. In these situations, 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. Sometimes broadcasting can be used, but using broadcasting to inform 1000 machines on a million-node network is inefficient because most receivers are not interested in the message (or worse yet, they are definitely interested but are not supposed to see it). 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. In this section we will describe one way of doing multicast routing. For additional information.

Multicasting requires group management. Some way is needed to create and destroy groups, and to allow processes to join and leave groups. How these tasks are accomplished is not of concern to the routing algorithm. What is of concern is that 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 hosts 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 propagates through the subnet.

To do multicast routing, each router computes a spanning tree covering all other routers. For example, in Figure 4.12(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 Figure 4.12(b).

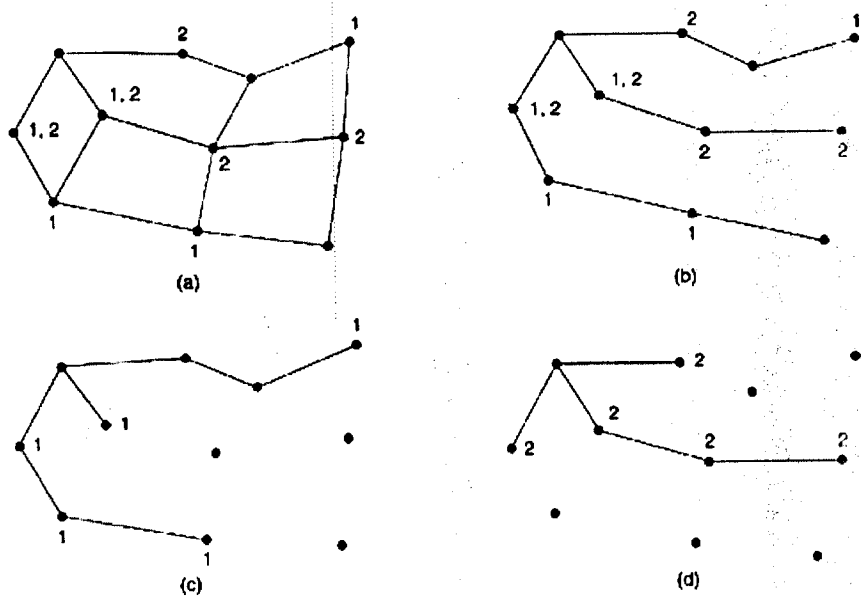


Figure 4.12 (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

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, Figure 4.12 (c) shows the pruned spanning tree for group 1. Similarly, Figure 4.12(d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning 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.

With distance vector routing, a different pruning strategy can be followed. The basic algorithm is reverse path forwarding. However, whenever a router with no hosts interested in a particular group and no connections to other routers receives a multicast message for that group, it responds with a PRUNE message, telling the sender not to send it any more multicasts for that group. When a router with no group members among its own hosts has received such messages on all its lines, it, too, can respond with a PRUNE message. In this way, the subnet is recursively pruned.

One potential disadvantage of this algorithm is that it scales poorly to large networks. Suppose that a network has n groups, each with an average of m members. For each group, m pruned spanning trees must be stored, for a total of mn trees. When many large groups exist, considerable storage is needed to store all the trees.

An alternative design uses core-based trees. Here, a single spanning tree per group is computed, with the root (the core) near the middle of the group. To send a multicast message, a host sends it to the core, which then does the multicast along the spanning tree. Although this tree will not be optimal for all sources, the reduction in storage costs from m trees to one tree per group is a major saving.

4.11 Routing for Mobile Hosts

Millions of people have portable computers nowadays, and they generally want to read their e-mail and access their normal file systems wherever in the world they may be. These mobile hosts introduce a new complication: to route a packet to a mobile host, the network first has to find it. The subject of incorporating mobile hosts into a network is very young, but in this section we will sketch some of the issues and give a possible solution.

The model of the world that network designers typically use is shown in Figure 4.13. Here we have a WAN consisting of routers and hosts.

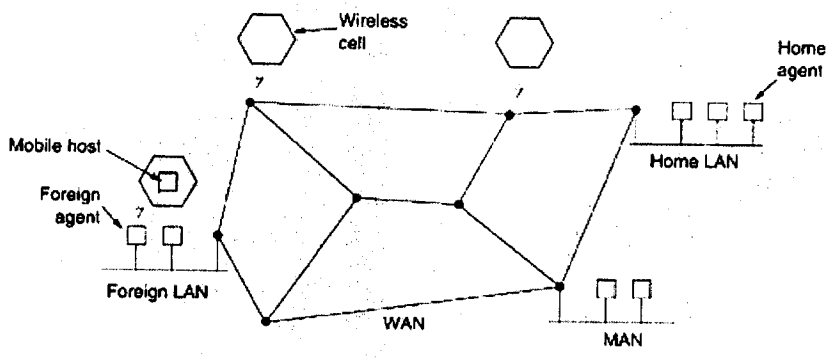


Figure 4.13 A WAN to which LANs, MANs, and wireless cells are attached.

Hosts that never move are said to be stationary. They are connected to the network by copper wires or fiber optics. In contrast, we can distinguish two other kinds of hosts. Migratory hosts are basically stationary hosts who move from one fixed site to another from time to time but use the network only when they are physically connected to it. Roaming hosts actually compute on the run and want to maintain their connections as they move around. We will use the term mobile hosts to mean either of the latter two categories, that is, all hosts that are away from home and still want to be connected.

All hosts are assumed to have a permanent home location that never changes. Hosts also have a permanent home address that can be used to determine their home locations, analogous to the way the telephone number 1-212-5551212 indicates the United States (country code 1) and Manhattan (212). The routing goal in systems with mobile hosts is to make it possible to send packets to mobile hosts using their home addresses and have the packets efficiently reach them wherever they may be. The trick, of course, is to find them.

In the model of Figure 4.13, 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 into the cell, his computer must register itself with the foreign agent there. The registration procedure typically works like this:

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

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

3. The foreign agent contacts the mobile host's home agent and says: One of your hosts is over here. The message from the foreign agent to the home agent contains the foreign agent's network address. It also includes the security information 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.

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 Figure 4.14. Here the sender, in the northwest city of Seattle, wants to send a packet to a host normally across the United States in New York. Packets sent to the mobile host on its home LAN in New York are intercepted by the home agent there. The home agent then looks up the mobile host's new (temporary) location and finds the address of the foreign agent handling the mobile host, in Los Angeles.

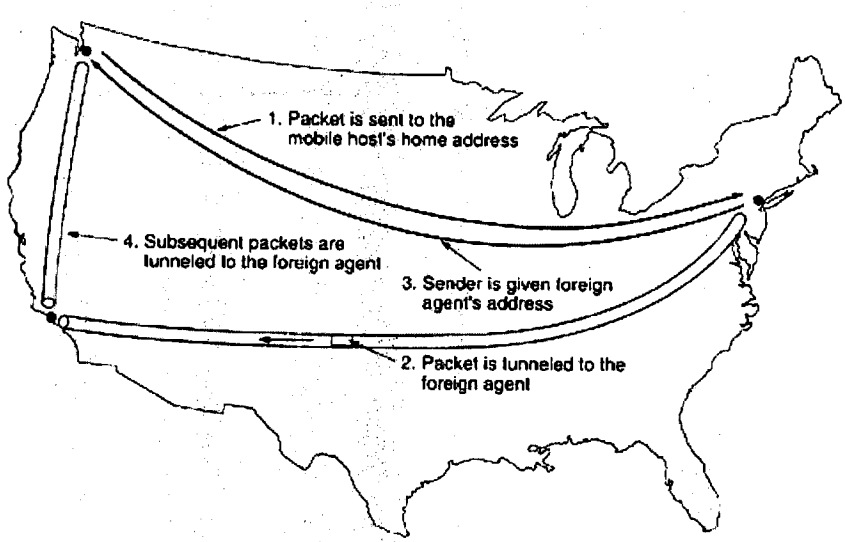


Figure 4.14. Packet routing for mobile hosts.

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 in Figure 4.14). This mechanism is called tunneling; we will look at it in more detail later. After getting the encapsulated packet, the foreign agent removes the original packet from the payload field and sends it to the mobile host as a data link frame.

Second, the home agent tells the sender to henceforth send packets to the mobile host by encapsulating them in the payload of packets explicitly addressed to the foreign agent instead of just sending them to the mobile host's home address (step 3). Subsequent packets can now be routed directly to the host via the foreign agent (step 4), bypassing the home location entirely.

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

Fourth, the schemes differ in how they actually manage to arrange for packets that are addressed to one destination to be delivered to a different one. One choice is changing the destination address and just retransmitting the modified packet. Alternatively, the whole packet, home address and all, can be encapsulated inside the payload of another packet sent to the temporary address. Finally, the schemes differ in their security aspects. In general, when a host or router gets a message of the form "Starting right now, please send all of Stephany's mail to me," it might have a couple of questions about whom it was talking to and whether this is a good idea.

4.12 Routing in Ad Hoc Networks

We have now seen how to do routing when the hosts are mobile but the routers are fixed. An even more extreme case is one in which the 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). Let us now examine them briefly. What makes ad hoc networks different from wired networks is that all the usual rules about fixed topologies, fixed and known neighbors, fixed relationship between IP address and location, and more are suddenly tossed out the window. Routers can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid indefinitely (barring a failure somewhere in the system). With an ad hoc network, the topology may be changing all the time, so desirability and even validity of paths can change spontaneously, without warning. Needless to say, these circumstances make routing in ad hoc networks quite different from routing in their fixed counterparts.

A variety of routing algorithms for ad hoc networks have been proposed. One of the more interesting ones is the AODV (Ad hoc 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. Another unusual characteristic is that 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. Let us now see what that means.

Route Discovery

At any instant of time, an ad hoc network can be described by a graph of the nodes (routers + hosts). Two nodes are connected (i.e., have an arc between them in the graph) if they can communicate directly using their radios. Since one of the two may have a more powerful transmitter than the other, it is possible that A is connected to B but B is not connected to A. However, for simplicity, we will assume all connections are symmetric. It should also be noted that the mere fact that two nodes are within radio range of each other does not mean that they are connected. There may be buildings, hills, or other obstacles that block their communication.

To describe the algorithm, consider the ad hoc network of Figure 4.15, 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 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."

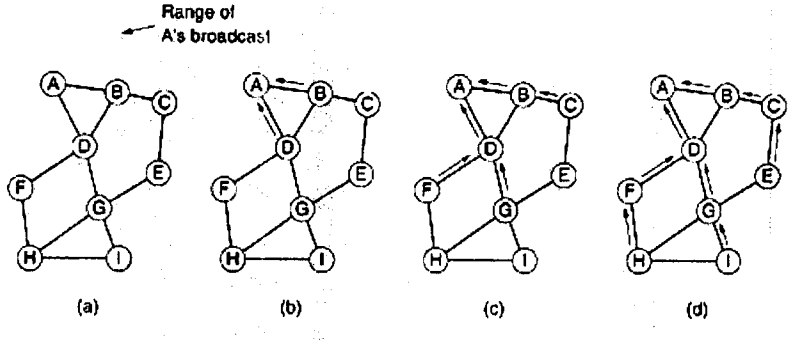


Figure 4.15. (a) Range of A's broadcast. (b) After B and D have received A's broadcast. (c) After C, F, and G have received A's broadcast. (d) After E, H, and I have received A's broadcast. The shaded nodes are new recipients. The arrows show the possible reverse routes.

To locate I, A constructs a special ROUTE REQUEST packet and broadcasts it. The packet reaches B and D, as illustrated in Figure 4.15(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.

The format of the ROUTE REQUEST packet is shown in Figure 4.16. It contains the source and destination addresses, typically their IP addresses, which identify that is looking for whom. It also contains a Request ID, which is a local counter maintained separately by each node and incremented each time a ROUTE REQUEST is broadcast. Together, the Source address and Request ID fields uniquely identify the ROUTE REQUEST packet to allow nodes to discard any duplicates they may receive.

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Figure 4.16 Format of a ROUTE REQUEST packet

In addition to the Request ID counter, each node also maintains a second sequence counter incremented whenever a ROUTE REQUEST is sent (or a reply to someone else's ROUTE REQUEST). It functions a little bit like a clock and is used to tell new routes from old routes. The fourth field of Figure 4.16 is A's sequence counter; the fifth field is the most recent value of I's sequence number that A has seen (0 if it has never seen it). The use of these fields will become clear shortly. The final field, Hop count, will keep track of how many hops the packet has made. It is initialized to 0.

When a ROUTE REQUEST packet arrives at a node (B and D in this case), it is processed in the following steps.

1. The (Source address, Request ID) pair is looked up in a local history table to see if this request has already been seen and processed. If it is a duplicate, it is discarded and processing stops. If it is not a duplicate, the pair is entered into the history table so future duplicates can be rejected, and processing continues.

2. The receiver looks up the destination in its route table. If a fresh route to the destination is known, a ROUTE REPLY packet is sent back to the source telling it how to get to the destination (basically: Use me). Fresh means that the Destination sequence number stored in the routing table is greater than or equal to the Destination sequence number in the ROUTE REQUEST packet. If it is less, the stored route is older than the previous route the source had for the destination, so step 3 is executed.

3. Since the receiver does not know a fresh route to the destination, it increments the Hop count field and rebroadcasts the ROUTE REQUEST packet. It also extracts the data from the packet and stores it as a new entry in its reverse route table. This information will be used to construct the reverse route so that the reply can get back to the source later. The arrows in Figure 4.15 are used for

building the reverse route. A timer is also started for the newly-made reverse route entry. If it expires, the entry is deleted.

Neither B nor D knows where I is, so each of them creates a reverse route entry pointing back to A, as shown by the arrows in Figure 4.15, and broadcasts the packet with Hop count set to 1. The broadcast from B reaches C and D. C makes an entry for it in its reverse route table and rebroadcasts it. In contrast, D rejects it as a duplicate. Similarly, D's broadcast is rejected by B. However, D's broadcast is accepted by F and G and stored, as shown in Figure 4.15(c). After E, H, and I receive the broadcast, the ROUTE REQUEST finally reaches a destination that knows where I is, namely, I itself, as illustrated in Figure 4.15(d). Note that although we have shown the broadcasts in three discrete steps here, the broadcasts from different nodes are not coordinated in any way.

In response to the incoming request, I build a ROUTE REPLY packet, as shown in Figure 4.17 The Source address, Destination address, and Hop count are copied from the incoming request, but the Destination sequence number taken from its counter in memory. The Hop count field is set to 0. The Lifetime field controls how long the route is valid. This packet is unicast to the node that the ROUTE REQUEST packet came from, in this case, G. It then follows the reverse path to D and finally to A. At each node, Hop count is incremented so the node can see how far from the destination (I) it is.

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Figure 4.17 Format of a ROUTE REPLY packet.

At each intermediate node on the way back, the packet is inspected. It is entered into the local routing table as a route to I if one or more of the following three conditions are met:

1. No route to I is known.
2. The sequence number for I in the ROUTE REPLY packet is greater than the value in the routing table.
3. The sequence numbers are equal but the new route is shorter.

In this way, all the nodes on the reverse route learn the route to I for free, as a byproduct of A's route discovery. Nodes that got the original REQUEST ROUTE packet but were not on the reverse path (B, C, E, F, and H in this example) discard the reverse route table entry when the associated timer expires.

In a large network, the algorithm generates many broadcasts, even for destinations that are close by. The number of broadcasts can be reduced as follows. The IP packet's Time to live is initialized by the sender to the

expected diameter of the network and decremented on each hop. If it hits 0, the packet is discarded instead of being broadcast.

The discovery process is then modified as follows. To locate a destination, the sender broadcasts a ROUTE REQUEST packet with Time to live set to 1. If no response comes back within a reasonable time, another one is sent, this time with Time to live set to 2. Subsequent attempts use 3, 4, 5, etc. In this way, the search is first attempted locally, then in increasingly wider rings.

Route Maintenance

Because nodes can move or be switched off, the topology can change spontaneously. For example, in Figure 4.16, if G is switched off, A will not realize that the route it was using to I (ADGI) is no longer valid. The algorithm needs to be able to deal with this. 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.

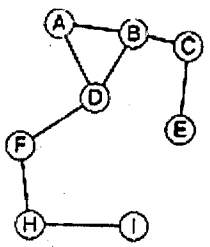
This information is used to purge routes that no longer work. For each possible destination, each node, N, keeps track of its neighbors that have fed it a packet for that destination during the last DT seconds. These are called N's active neighbors for that destination. N does this by having a routing table keyed by destination and containing the outgoing node to use to reach the destination, the hop count to the destination, the most recent destination sequence number, and the list of active neighbors for that destination. A possible routing table for node D in our example topology is shown in Figure 4.18(a).

When any of N's neighbors becomes unreachable, it checks its routing table to see which destinations have routes using the now-gone neighbor. For each of these routes, the active neighbors are informed that their route via N is now invalid and must be purged from their routing tables. The active neighbors then tell their active neighbors, and so on, recursively, until all routes depending on the now-gone node are purged from all routing tables.

As an example of route maintenance, consider our previous example, but now with G suddenly switched off. The changed topology is illustrated in Figure 4.18(b). When D discovers that G is gone, it looks at its routing table and sees that G was used on routes to E, G, and I. The union of the active neighbors for these destinations is the set {A, B}. In other words, A and B depend on G for some of their routes, so they have to be informed that these routes no longer work. D tells them by sending them packets that cause them to update their own routing tables accordingly. D also purges the entries for E, G, and I from its routing table.

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Figure 4.18 (a) D's routing table before G goes down. (b) The graph after G has gone down.

It may not have been obvious from our description, but a critical difference between AODV and Bellman-Ford is that nodes do not send out periodic broadcasts containing their entire routing table. This difference saves both bandwidth and battery life.

Node Lookup in Peer-to-Peer Networks

A relatively new phenomenon is peer-to-peer networks, in which a large number of people, usually with permanent wired connections to the Internet, are in contact to share resources. The first widespread application of peer-to-peer technology was for mass crime: 50 million Napster users were exchanging copyrighted songs without the copyright owners' permission until Napster was shut down by the courts amid great controversy. Nevertheless, peer-to-peer technology has many interesting and legal uses. It also has something similar to a routing problem, although it is not quite the same as the ones we have studied so far. Nevertheless, it is worth a quick look.

What makes peer-to-peer systems interesting is that they are totally distributed. All nodes are symmetric and there is no central control or hierarchy. In a typical peer-to-peer system the users each have some information that may be of interest to other users. This information may be free software, (public domain) music, photographs, and so on. If there are large numbers of users, they will not know each other and will not know where to find what they are looking for. One solution is a big central database, but this may not be feasible for some reason (e.g., nobody is willing to host and maintain it). Thus, the problem comes down to how a user finds a node that contains what he is looking for in the absence of a centralized database or even a centralized index.

Let us assume that each user has one or more data items such as songs, photographs, programs, files, and so on that other users might want to read. Each item has an ASCII string naming it. A potential user knows just the ASCII string and wants to find out if one or more people have copies and, if so, what their IP addresses are.

As an example, consider a distributed genealogical database. Each genealogist has some on-line records for his or her ancestors and relatives, possibly with photos, audio, or even video clips of the person. Multiple people may have the same great grandfather, so an ancestor may have records at multiple nodes. The name of the record is the person's name in some canonical form. At some point, a genealogist discovers his great grandfather's will in an archive, in which the great grandfather bequeaths his gold pocket watch to his nephew. The genealogist now knows the nephew's name and wants to find out if any other genealogist has a record for him. How, without a central database, do we find out who, if anyone, has records?

Various algorithms have been proposed to solve this problem. A simplified explanation of how it works is as follows. The Chord system consists of n participating users, each of whom may have some stored records and each of whom is prepared to store bits and pieces of the index for use by other users. Each user node has an IP address that can be hashed to an m-bit number using a hash function, hash. Chord uses SHA-1 for hash. For now, it is just a function that takes a variable-length byte string as argument and produces a highly-random 160-bit number. Thus, we can convert any IP address to a 160-bit number called the node identifier.

Conceptually, all the 2^{160} node identifiers are arranged in ascending order in a big circle. Some of them correspond to participating nodes, but most of them do not. In Figure 4.19(a) we show the node identifier circle for m = 5 (just ignore the arcs in the middle for the moment). In this example, the nodes with identifiers 1, 4, 7, 12, 15, 20, and 27 correspond to actual nodes and are shaded in the figure; the rest do not exist.

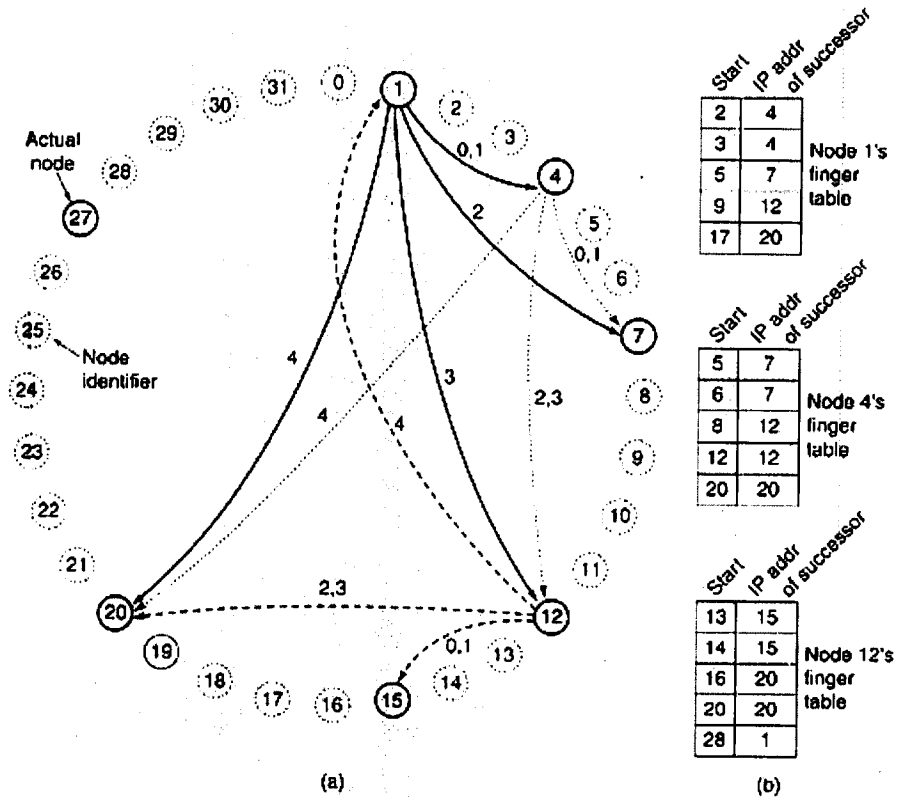


Figure 4.19(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4, and 12. The labels on the arcs are the table indices. (b) Examples of the finger tables.

Let us now define the function successor(k) as the node identifier of the first actual node following k around the circle clockwise. For example, successor (6) = 7, successor (8) = 12, and successor (22) = 27.

The names of the records (song names, ancestors' names, and so on) are also hashed with hash (i.e., SHA-1) to generate a 160-bit number, called the key. Thus, to convert name (the ASCII name of the record) to its key, we use key = hash(name). This computation is just a local procedure call to hash. If a person holding a genealogical record for name wants to make it available to everyone, he first builds a tuple consisting of (name, my-IP-address) and then asks successor(hash(name)) to store the tuple. If multiple records (at different nodes) exist for this name, their tuple will all be stored at the same node. In this way, the index is distributed over the nodes at random. For fault tolerance, p different hash functions could be used to store each tuple at p nodes, but we will not consider that further here.

If some user later wants to look up name, he hashes it to get key and then uses successor (key) to find the IP address of the node storing its index

tuples. The first step is easy; the second one is not. To make it possible to find the IP address of the node corresponding to a certain key, each node must maintain certain administrative data structures. One of these is the IP address of its successor node along the node identifier circle. For example, in Figure 4.19, node 4's successor is 7 and node 7's successor is 12.

Lookup can now proceed as follows. The requesting node sends a packet to its successor containing its IP address and the key it is looking for. The packet is propagated around the ring until it locates the successor to the node identifier being sought. That node checks to see if it has any information matching the key, and if so, returns it directly to the requesting node, whose IP address it has.

As a first optimization, each node could hold the IP addresses of both its successor and its predecessor, so that queries could be sent either clockwise or counterclockwise, depending on which path is thought to be shorter. For example, node 7 in Figure 4.19 could go clockwise to find node identifier 10 but counterclockwise to find node identifier 3.

Even with two choices of direction, linearly searching all the nodes is very inefficient in a large peer-to-peer system since the mean number of nodes required per search is n/2. To greatly speed up the search, each node also maintains what Chord calls a finger table. The finger table has m entries, indexed by 0 through m - 1, each one pointing to a different actual node. Each of the entries has two fields: start and the IP address of successor(start), as shown for three example nodes in Figure 4.19(b). The values of the fields for entry i at node k are:

$$start = k + 2^i \ (modulo\, 2^m)$$
IP address of $successor(start[i])$

Note that each node stores the IP addresses of a relatively small number of nodes and that most of these are fairly close by in terms of node identifier.

Using the finger table, the lookup of key at node k proceeds as follows. If key falls between k and successor (k), then the node holding information about key is successor (k) and the search terminates. Otherwise, the finger table is searched to find the entry whose start field is the closest predecessor of key. A request is then sent directly to the IP address in that finger table entry to ask it to continue the search. Since it is closer to key but still below it, chances are good that it will be able to return the answer with only a small number of additional queries. In fact, since every lookup halves the remaining distance to the target, it can be shown that the average number of lookups is $\log_2 n$.

As a first example, consider looking up key = 3 at node 1. Since node 1 knows that 3 lies between it and its successor, 4, the desired node is 4 and the search terminates, returning node 4's IP address.

As a second example, consider looking up key = 14 at node 1. Since 14 does not lie between 1 and 4, the finger table is consulted. The closest

predecessor to 14 is 9, so the request is forwarded to the IP address of 9's entry, namely, that of node 12. Node 12 sees that 14 falls between it and its successor (15), so it returns the IP address of node 15.

As a third example, consider looking up key = 16 at node 1. Again a query is sent to node 12, but this time node 12 does not know the answer itself. It looks for the node most closely preceding 16 and finds 14, which yields the IP address of node 15. A query is then sent there. Node 15 observes that 16 lies between it and its successor (20), so it returns the IP address of 20 to the caller, which works its way back to node 1.

Since nodes join and leave all the time, Chord needs a way to handle these operations. We assume that when the system began operation it was small enough that the nodes could just exchange information directly to build the first circle and finger tables. After that an automated procedure is needed, as follows. When a new node, r, wants to join, it must contact some existing node and ask it to look up the IP address of successor (r) for it. The new node then asks successor (r) for its predecessor. The new node then asks both of these to insert r in between them in the circle. For example, if 24 in Figure 4.19 wants to join, it asks any node to look up successor (24), which is 27. Then it asks 27 for its predecessor (20). After it tells both of those about its existence, 20 uses 24 as its successor and 27 uses 24 as its predecessor. In addition, node 27 hands over those keys in the range 21–24, which now belong to 24. At this point, 24 is fully inserted.

However, many finger tables are now wrong. To correct them, every node runs a background process that periodically recomputes each finger by calling successor. When one of these queries hits a new node, the corresponding finger entry is updated.

When a node leaves gracefully, it hands its keys over to its successor and informs its predecessor of its departure so the predecessor can link to the departing node's successor. When a node crashes, a problem arises because its predecessor no longer has a valid successor. To alleviate this problem, each node keeps track not only of its direct successor but also its s direct successors, to allow it to skip over up to s - 1 consecutive failed nodes and reconnect the circle.

4.13 File Transfer Protocol (FTP)

The main task of the Internet is to provide services for users. Among the most popular applications are FTP, DNS, DHCP, SNMP, SMTP, HTML, HTTP, RPC, and Middleware.

Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to tile transfer. In this section, we discuss one popular protocol involved in transferring tiles: File Transfer Protocol (FTP).

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the standard mechanism 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. For example, two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. All these problems have been solved by FTP in a very simple and elegant approach.

FTP differs from other client/server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). 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. However, the difference in complexity is at the FTP level, not TCP. For TCP, both connections are treated the same.

FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.

Figure 4.20 shows the basic model of FTP. The client has three components: user interface, client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.

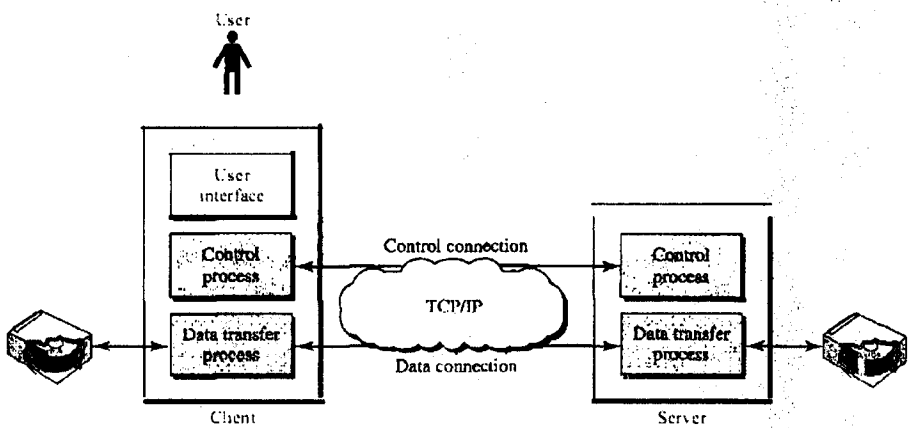


Figure 4.20 FTP

The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each tile transferred. It opens each time commands that involve transferring files are

used, and it closes when the tile is transferred. In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several tiles are transferred.

Communication over Control Connection

FTP uses the same approach as SMTP to communicate across the control connection. It uses the 7-bit ASCII character set (see Figure 4.21). Communication is achieved through commands and isponses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line, so we need not worry about file format or file structure. Each line is terminated with a two-character (carriage return and line teed) end of-line token.

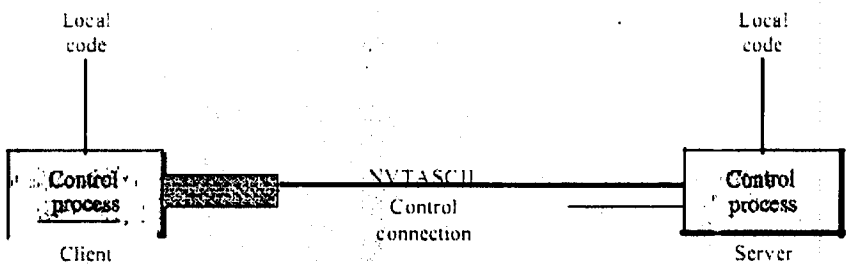


Figure 4.21 Using the control connection

Communication over Data Connection

The purpose of the data connection is different from that of the control connection. We want to transfer tiles through the data connection. File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things:

- A file is to be copied from the server to the client. This is called retrieving aft/e. It is done under the supervision of the RETR command,
- A file is to be copied from the client to the server. This is called storing aft/e. It is done under the supervision of the STOR command.
- A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command.

Note that FTP treats a list of directory or file names as a file. It is sent over the data connection.

The client must define the type of file to be transferred, the structure of the data, and the transmission mode. Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: tile type, data structure, and transmission mode (see Figure 4.22).

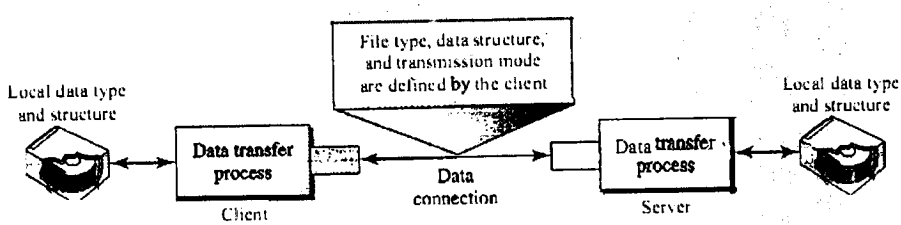


Figure 4.22 Using the data connection

File Type FTP can transfer one of the following file types across the data connection: an ASCII file, EBCDIC tile, or image file. The ASCII file is the default format for transferring text files. Each character is encoded using 7-bit ASCII. The sender transforms the file from its own representation into ASCII characters, and the receiver transforms the ASCII characters to its own representation. If one or both ends of the connection use EBCDIC encoding (the file format used by IBM), the file can be transferred using EBCDIC encoding. The image tile is the default format for transferring binary files. The tile is sent as continuous streams of bits without any interpretation or encoding. This is mostly used to transfer binary tiles such as compiled programs.

Data Structure FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: file structure, record structure, and page structure. In the file structure format, the file is a continuous stream of bytes. In the record structure, the file is divided into records. This can be used only with text tiles. In the page structure, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Transmission Mode FTP can transfer a file across the data connection by using one of the following three transmission modes: stream mode, block mode, and compressed mode. The stream mode is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data

are simply a stream of bytes (file structure), no end-of-file is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a l-byte end- of-record (EOR) character and the end of the tile will have a l-byte end-of-file (EOF) character. In block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the block descriptor; the next 2 bytes define the size of the block in bytes. In the compressed mode, if the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the number of repetitions. In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed.

4.14 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 4.23).

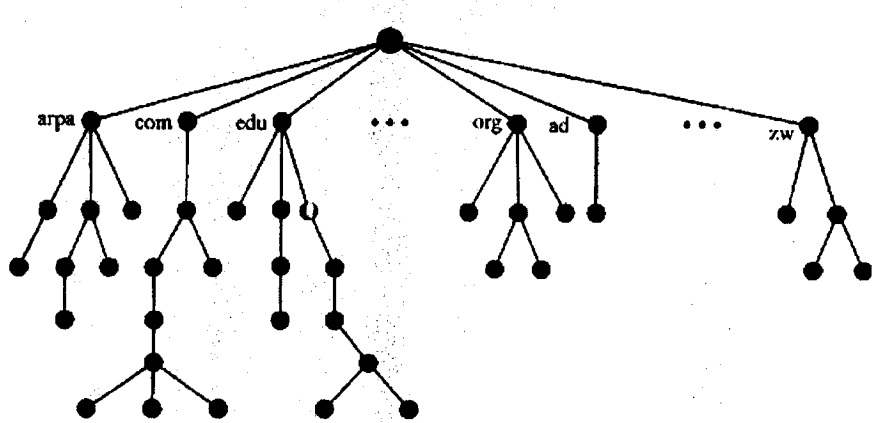


Figure 4.23 Domain name space

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 frill 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 4.24 shows some domain names.

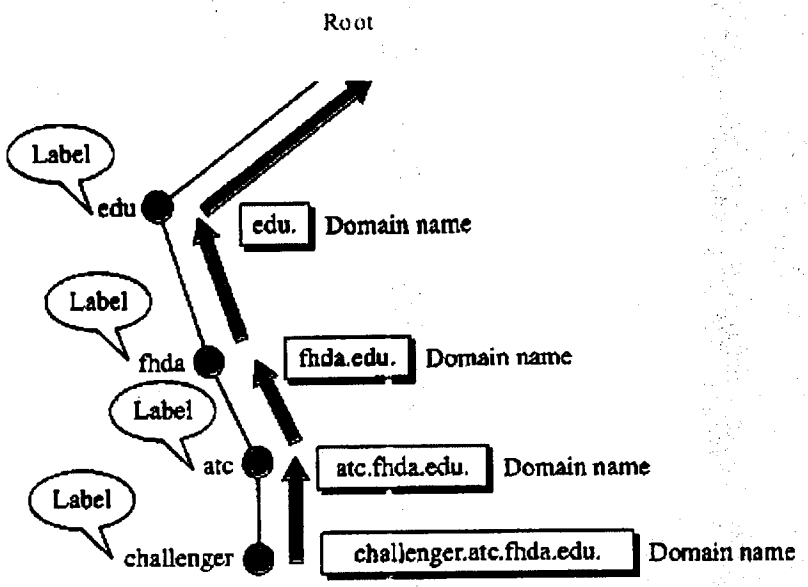


Figure 4.24 Domain names and labels

Fully Qualified Domain Name

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name

challenger, ate. tbda. edu.

is the FQDN of a computer named challenger installed at the Advanced Technology Center (ATC) at De Anza College. A DNS server can only match an FQDN to an address. Note that the name must end with a null label, but because null means nothing, the label ends with a dot (.).

Partially Qualified Domain Name

If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN. For example, if a user at the jhda.edu. site wants to get the IP address of the challenger computer, he or she can define the partial name

challenger

The DNS client adds the suffix atc.jhda.edu. before passing the address to the DNS server. The DNS client normally holds a list of suffixes. The following can be the list of suffixes at De Anza College. The null suffix defines nothing. This suffix is added when the user defines an FQDN.

atc.fhda.edu

fhda. edu

null

Figure 4.25 shows some FQDNs and PQDNs.

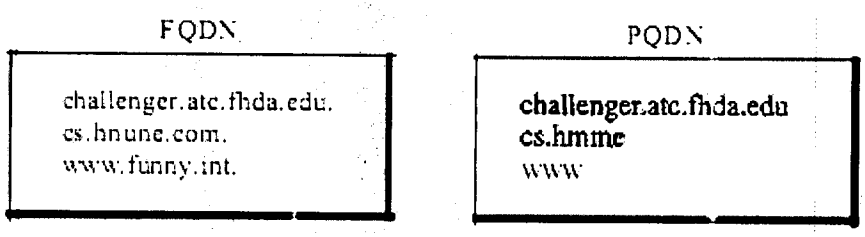


Figure 4.25 FQDN and PQDN

Domain

A domain is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Figure 4.26 shows some domains. Note that a domain may itself be divided into domains (or subdomains as they are sometimes called).

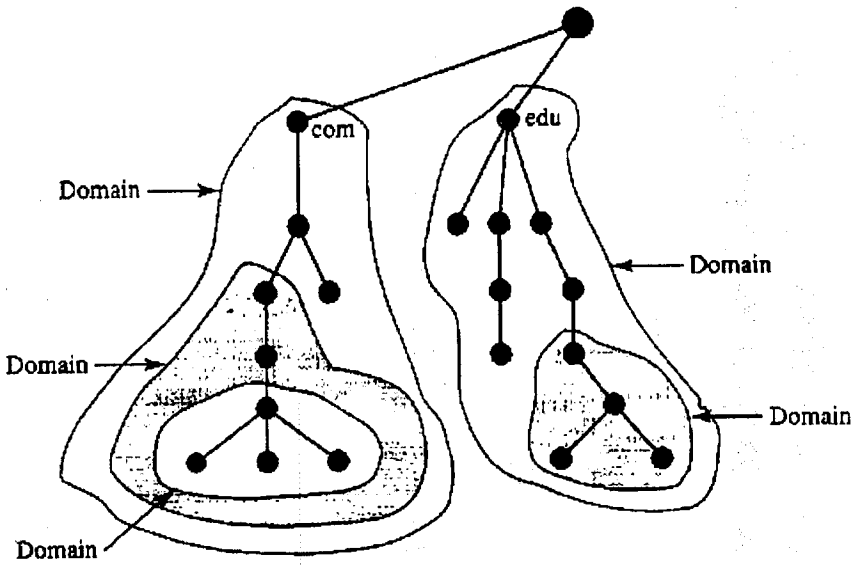


Figure 4.26 Domains

4.15 Dynamic Host Configuration Protocol (DHCP)

BOOTP is not a dynamic configuration protocol. When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address. This implies that the binding between the physical address and the IP address of the client already exists. The binding is predetermined.

However, what if a host moves from one physical network to another? What if a host wants a temporary IP address? BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator. BOOTP is a static configuration protocol.

The Dynamic Host Configuration Protocol (DHCP) has been devised to provide static and dynamic address allocation that can be manual or automatic.

Static Address Allocation In this capacity DHCP acts as BOOTP does. It is backward- compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.

Dynamic Address Allocation DHCP has a second database with a pool of available IP addresses. This second database makes DHCP dynamic.

When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

When a DHCP client sends a request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned. On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.

The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network (as is a subscriber to a service provider). DHCP provides temporary IP addresses for a limited time.

The addresses assigned from the pool are temporary addresses. The DHCP server issues a lease for a specific time. When the lease expires, the client must either stop using the IP address or renew the lease. The server has the option to agree or disagree with the renewal. If the server disagrees, the client stops using the address.

Manual and Automatic Configuration One major problem with the BOOTP protocol is that the table mapping the IP addresses to physical addresses needs to be manually configured. This means that every time there is a change in a physical or IP address, the administrator needs to manually enter the changes. DHCP, on the other hand, allows both manual and automatic configurations. Static addresses are created manually; dynamic addresses are created automatically.

4.16 Simple Network Management Protocol (SNMP)

We can define network management as monitoring, testing, configuring, and troubleshooting network components to meet a set of requirements defined by an organization. These requirements include the smooth, efficient operation of the network that provides the predefined quality of service for users. To accomplish this task, a network management system uses hardware, software, and humans. In this chapter, first we briefly discuss the functions of a network management system. Then we concentrate on the most common management system, the Simple Network Management Protocol (SNM P).

Network Management System

We can say that the functions performed by a network management system can be divided into five broad categories: configuration management, fault management, performance management, security management, and accounting management, as shown in Figure 4.27.

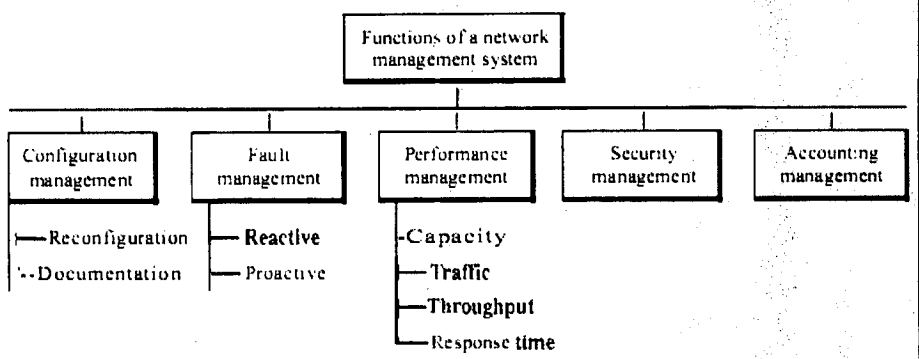


Figure 4.27 Functions of a network management system

Configuration Management

A large network is usually made up of hundreds of entities that are physically or logically connected to one another. These entities have an initial configuration when the network is set tip, but can change with time. Desktop computers may be replaced by others; application software may be updated to a newer version; and users may move from one group to another. The configuration management system must know, at any time, the status of each entity and its relation to other entities. Configuration management can be divided into two subsystems: reconfiguration and documentation.

Reconfiguration

Reconfiguration, which means adjusting the network components and features, can be a daily occurrence in a large network. There are three types of reconfiguration: hardware reconfiguration, software reconfiguration, and user-account reconfiguration.

Hardware reconfiguration covers all changes to the hardware. For example, a desktop computer may need to be replaced. A router may need to be moved to another part of the network. A subnet work may be added or removed from the network. All these need the time and attention of network management. In a large network, there must be specialized personnel trained for quick and efficient hardware reconfiguration. Unfortunately, this

type of reconfiguration cannot be automated and must be manually handled case by case.

Software reconfiguration covers all changes to the software. For example, new software may need to be installed on servers or clients. An operating system may need updating. Fortunately, most software reconfiguration can be automated. For example, updating an application on some or all clients can be electronically downloaded from the server.

User-account reconfiguration is not simply adding or deleting users on a system. You must also consider the user privileges, both as an individual and as a member of a group. For example, a user may have read and write permission with regard to some files, but only read permission with regard to other files. User-account reconfiguration can be, to some extent, automated. For example, in a college or university, at the beginning of each quarter or semester, new students are added to the system. The students are normally grouped according to the courses they take or the majors they pursue.

Documentation

The original network configuration and each subsequent change must be recorded meticulously. This means that there must be documentation for hardware, software, and user accounts.

Hardware documentation normally involves two sets of documents: maps and specifications. Maps track each piece of hardware and its connection to the network. There can be one general map that shows the logical relationship between each subnetwork. There can also be a second general map that shows the physical location of each subnetwork. For each subnetwork, then, there is one or more maps that show all pieces of equipment. The maps use some kind of standardization to be easily read and understood by current and future personnel. Maps are not enough per se. Each piece of hardware also needs to be documented. There must be a set of specifications for each piece of hardware connected to the network. These specifications must include information such as hardware type, serial number, vendor (address and phone number), time of purchase, and warranty information.

All software must also be documented. Software documentation includes information such as the software type, the version, the time installed, and the license agreement.

Most operating systems have a utility that allows the documentation of user accounts and their privileges. The management must make sure that the

files with this information are updated and secured. Some operating systems record access privileges in two documents; one shows all files and access types for each user; the other shows the list of users that have a particular access to a file.

Fault Management

Complex networks today are made up of hundreds and sometimes thousands of components. Proper operation of the network depends on the proper operation of each component individually and in relation to each other. Fault management is the area of network management that handles this issue.

An effective fault management system has two subsystems: reactive fault management and proactive fault management.

Reactive Fault Management

A reactive fault management system is responsible for detecting, isolating, correcting, and recording faults. It handles short term solutions to faults.

The first step taken by a reactive fault management system is to detect the exact location of the fault. A fault is defined as an abnormal condition in the system. When a fault occurs, either the system stops working properly or the system creates excessive errors. A good example of a fault is a damaged communication medium. This fault may interrupt communication or produce excessive errors.

The next step taken by a reactive fault management system is to isolate the fault. A fault, if isolated, usually affects only a few users. After isolation, the affected users are immediately notified and given an estimated time of correction.

The third step is to correct the fault. This may involve replacing or repairing the faulty component(s).

After the fault is corrected, it must be documented. The record should show the exact location of the fault, the possible cause, the action or actions taken to correct the fault, the cost, and time it took for each step. Documentation is extremely important for several reasons:

- The problem may recur. Documentation can help the present or future administrator or technician solve a similar problem.
- The frequency of the same kind of thilure is an indication of a major problem in the system. If a fault happens frequently in one

component, it should be replaced with a similar one, or the whole system should be changed to avoid the use of that type of component.

- The statistic is helpful to another part of network management, performance management.

Proactive Fault Management

Proactive fault management tries to prevent faults from occurring. Although this is not always possible, some types of failures can be predicted and prevented. For example, if a manufacturer specifies a lifetime for a component or a part of a component, it is a good strategy to replace it before that time. As another example, if a fault happens frequently at one particular point of a network, it is wise to carefully reconfigure the network to prevent the fault from happening again.

Performance Management

Performance management, which is closely related to fault management, tries to monitor and control the network to ensure that it is running as efficiently as possible. Performance management tries to quantify performance by using some measurable quantity such as capacity, traffic, throughput, or response time.

capacity

One factor that must be monitored by a performance management system is the capacity of the network. Every network has a limited capacity, and the performance management system must ensure that it is not used above this capacity. For example, if a LAN is designed for 100 stations at an average data rate of 2 Mbps, it will not operate properly if 200 stations are connected to the network. The data rate will decrease and blocking may occur.

Traffic

Traffic can be measured in two ways: internally and externally. Internal traffic is measured by the number of packets (or bytes) traveling inside the network. External traffic is measured by the exchange of packets (or bytes) outside the network. During peak hours, when the system is heavily used, blocking may occur if there is excessive traffic.

Throughput

We can measure the throughput of an individual device (such as a router) or a part of the network. Performance management monitors the throughput to make sure that it is not reduced to unacceptable levels.

Response Tim e

Response time is normally measured from the time a user requests a service to the time the service is granted. Other factors such as capacity and traffic can affect the response time. Performance management monitors the average response time and the peak-hour response time. Any increase in response time is a very serious condition as it is an indication that the network is working above its capacity.

Security Management

Security management is responsible for controlling access to the network based on the predefined policy.

Accounting Management

Accounting management is the control of user's access to network resources through charges. Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network. Charging does not necessarily mean cash transfer; it may mean debiting the departments or divisions for budgeting purposes. Today, organizations use an accounting management system for the following reasons:

- It prevents users from monopolizing limited network resources.
- It prevents users from using the system inefficiently.
- Network managers can do short and long term planning based on the demand for network use.

4.17 Electronic Mail (SMTP)

One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program. Its architecture consists of several components that we discuss in this chapter.

At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message

to include text, audio, and video. It also allows one message to be sent to one or more recipients.

In this chapter, we first study the general architecture of an e-mail system including the three main components: user agent, message transfer agent, and message access agent. We then describe the protocols that implement these components.

Architecture

To explain the architecture of e-mail, we give four scenarios. We begin with the simplest situation and add complexity as we proceed. The fourth scenario is the most common in the exchange of email.

First Scenario

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same system; they are directly connected to a shared system. The administrator has created one mailbox for each user where the received messages are stored. A mailbox is part of a local hard drive, a special tile with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, Alice runs a user agent (p4) program to prepare the message and store it in Bob's mailbox. The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience, using a user agent. Figure 4.28 shows the concept.

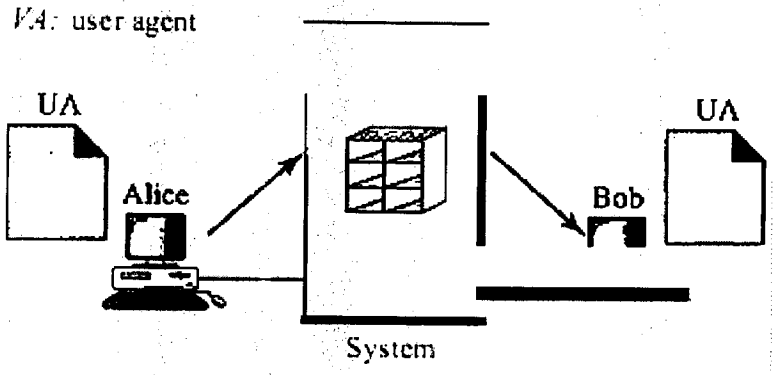


Figure 4.28 First scenario in electronic mail

This is similar to the additional memo exchange between employees in an office. There is a mailroom where each employee has a mailbox with his or her name on it.

When Alice needs to send a memo to Bob, she writes the memo and inserts it into Bob's mailbox. When Bob checks his mailbox, he finds Alice's memo and reads it.

Second Scenario

In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systerns. The message needs to be sent over the Internet. Here we need user agents (VAs) and message transfer agents (MTAs). as shown in Figure 4.29.

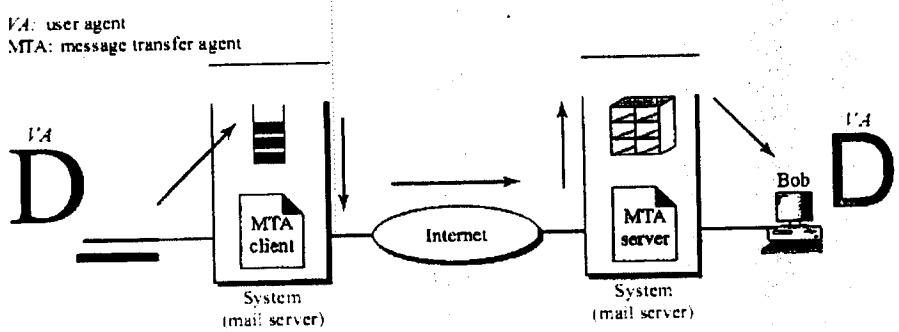


Figure 4.29 Second scenario in electronic mail

Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her site uses a queue to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. 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 alerted by the system when there is a message in the queue to be sent.

Third Scenario

In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for 1ndling e-mails-all users need to send their messages to this mail server. Figure 4.30 shows the situation.

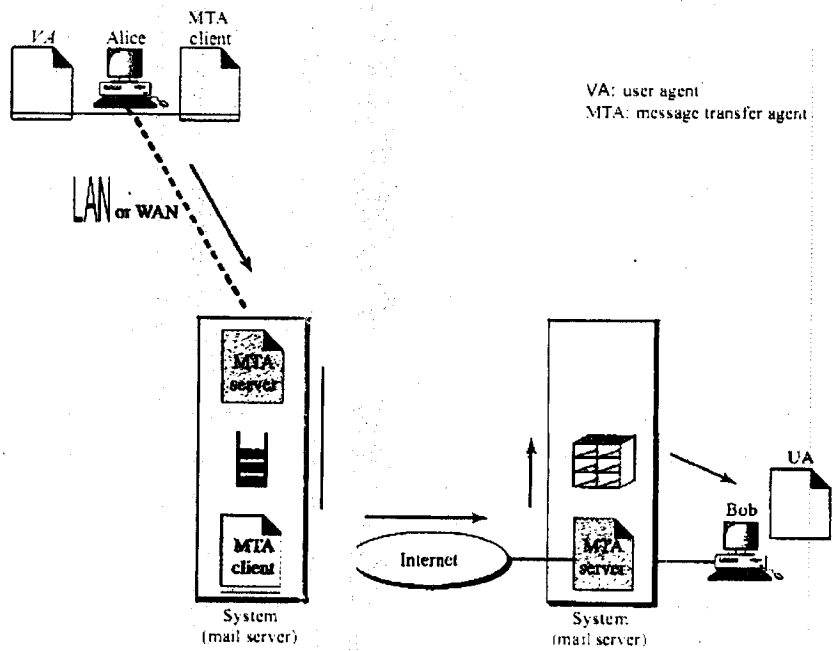


Figure 4.30 Third scenario in electronic mail

Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or \VAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in turn, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox.

At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client/server programs.

Fourth Scenario

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob reeds to retrieve

it. Here, we need another set of client/server agents, which we call message access agents (MAAs). Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages. The situation is shown in Figure 4.31.

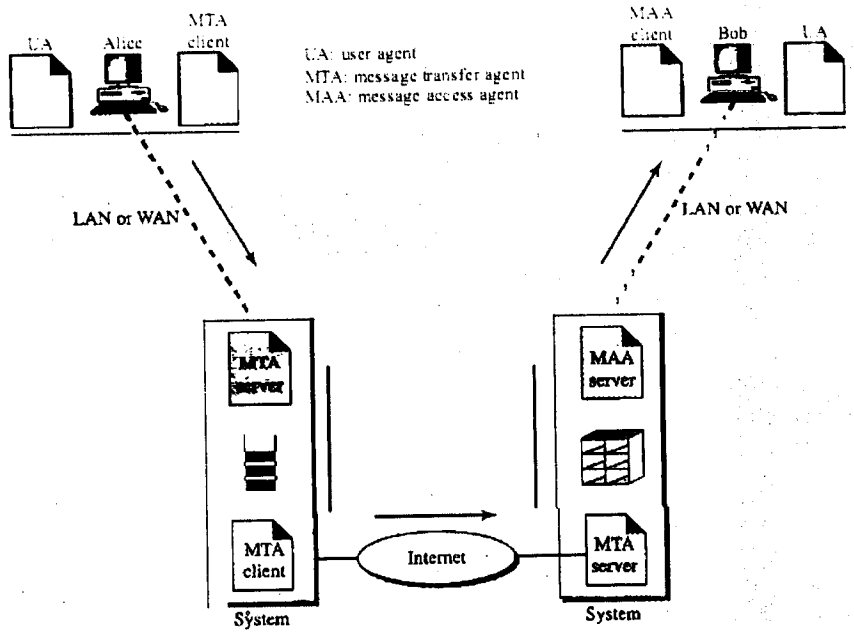


Figure 4.31 Fourth scenario in electronic mail

There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a-WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

Second, note that Bob needs another pair of client/server programs: message access programs. This is so because an MTA client/server program is push program: the client pushes the message to the server. Bob needs push program. The client needs to pull the message from the server. Figure 4.32 shows the difference.

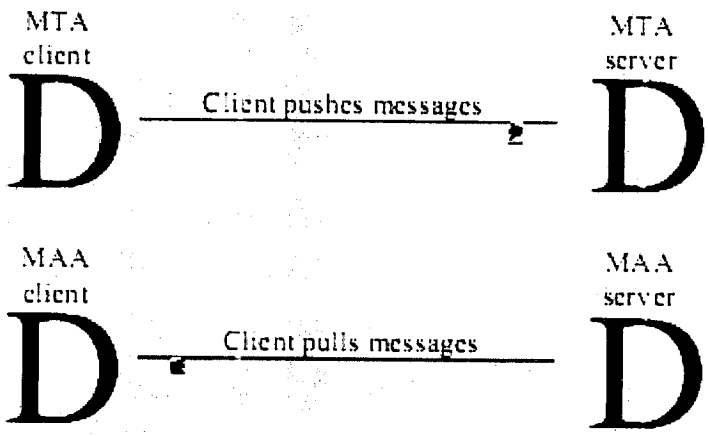


Figure 4.32 Push versus pull in electronic email

User Agent

The first component of an electronic mail system is the user agent (VA). It provides service to the user to make the process of sending and receiving a message easier.

Services Provided by a User Agent

A user agent is a software package (program) that composes, reads, replies to, and tar- wards messages. It also handles mailboxes. Figure 4.33 shows the services of a typical user agent.

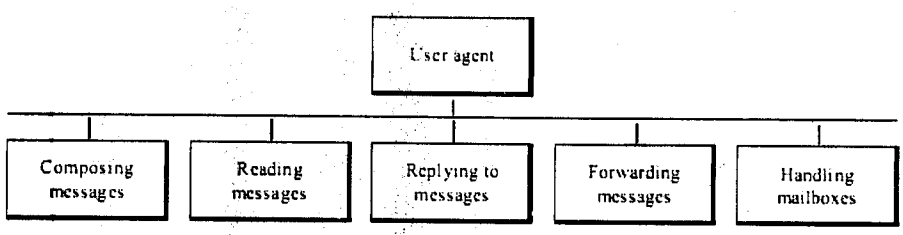


Figure 4.33 services of user agent

Composing Messages A user agent helps the user compose the e-mail message to be sent out. Most user agents provide a template on the screen to be filled in by the user. Some even have a built-in editor that can do spell checking, grammar checking, and other tasks expected from a sophisticated word processor. A user, of course, could alternatively use his or her favorite text editor or word processor to create the message and import it, or cut and paste it, into the user agent template.

Reading Messages The second duty of the user agent is to read the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Most user agents show a one-line summary of each received mail. Each e-mail contains the following fields.

1. A number field.

2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to.

3. The size of the message.

4. The sender.

5. The optional subject field.

Replying to Messages After reading a message, a user can use the user agent to reply to a message. A user agent usually allows the user to reply to the original sender or to reply to all recipients of the message. The reply message may contain the original message (for quick reference) and the new message.

Forwarding Messages Replying is defined as sending a message to the sender or recipients of the copy. Forwarding is defined as sending the message to a third party. A user agent allows the receiver to forward the message, with or without extra comments, to a third party.

Handling Mailboxes

A user agent normally creates two mailboxes: an inbox and an out box. Each box is a file with a special format that can be handled by the user agent. The inbox keeps all the received e-mails until they are deleted by the user. The out box keeps all the sent e-mails until the user deletes them. Most user agents today are capable of creating customized mailboxes.

User Agent Types

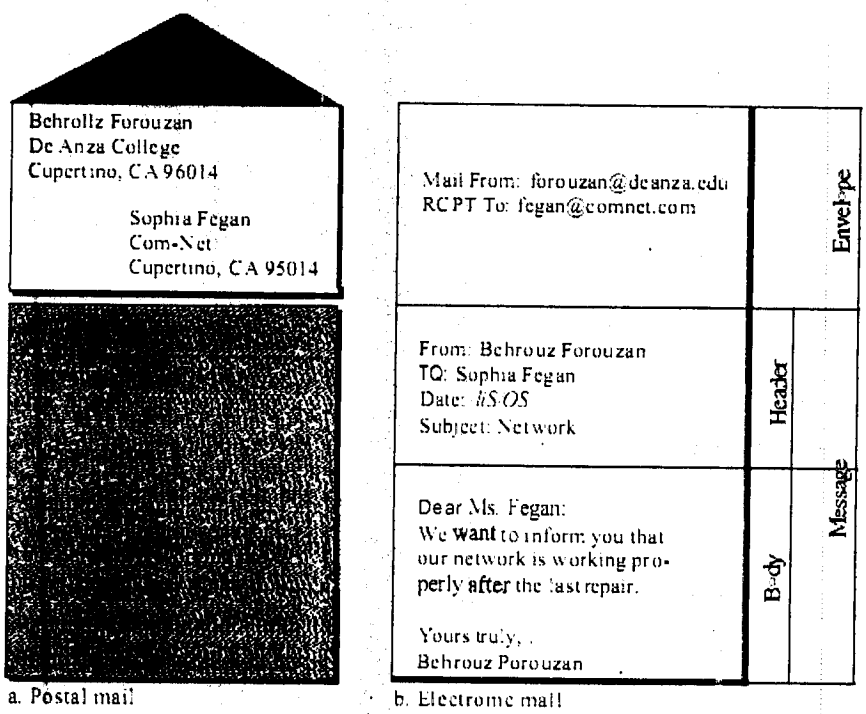
There are two types of user agents: command-driven and GUI-based.

Command-Driven Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients. Some examples of command-driven user agents are mail, pine, and elm.

GUI-Based Modern user agents are GUI-based. They contain graphical-user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu bars, and windows that make the services easy to access. Some examples of GUI-based user agents are Eudora, Microsoft's Outlook, and Netscape.

Sending Mail

To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message (see Figure 4.34).



a. Postal mail b. Electronic mail

Figure 4.34 Format of an e-mail

Envelope The envelope usually contains the sender and the receiver addresses.

Message The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information (such as encoding type, as we see shortly). The body of the message contains the actual information to be read by the recipient.

Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the VA informs the user with a notice. If the user is ready to read the mail a list is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a local part and a domain name, separated by an sign (see Figure 4.35).

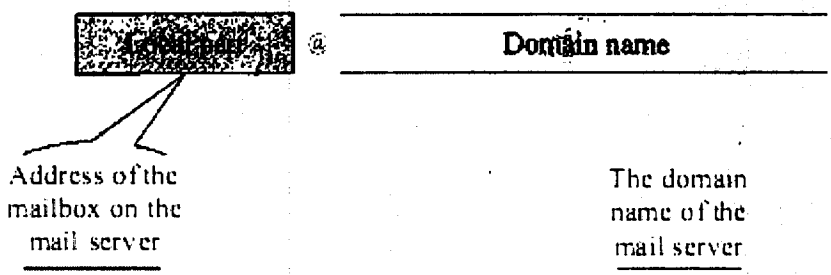


Figure 4.35 E-mail address

Local Part The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.

Domain Name The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; the hosts are sometimes called mail servers or exchangers. The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (for example, the name of the organization).

Mailing List

Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipients name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA. I f there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity.

4.18 World Wide Web (WWW)

The World Wide Web (WWW) is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research.

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, as shown in Figure 4.36.

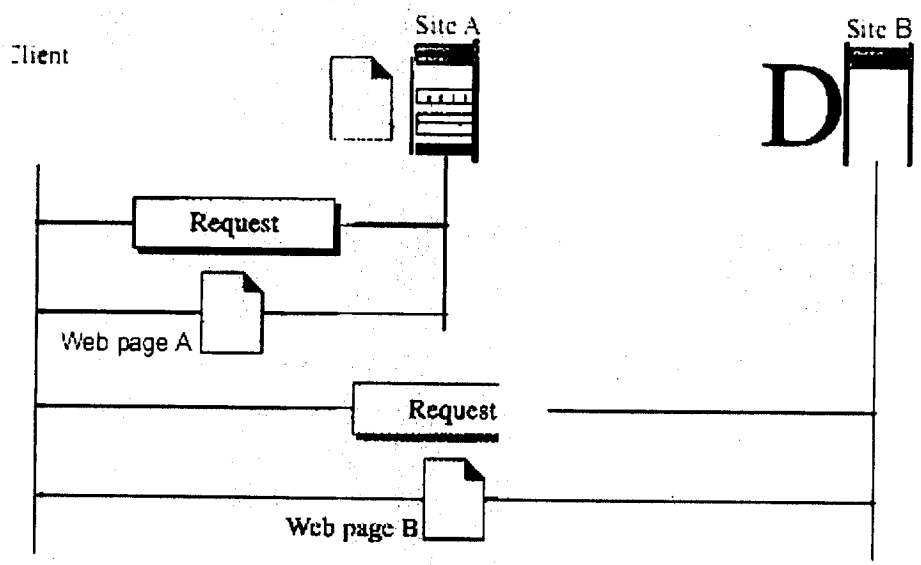


Figure 4.36 Architecture of WWW

Each site holds one or more documents, referred to as web pages. Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers. Let us go through the scenario shown in Figure 4.36. The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL. The server at site A finds the document and sends it to the client. When the user views the document, she finds some references to other documents, including a Web page at site B. The

reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. 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 previously such as FTP or HTIP (described later in the chapter). The interpreter can be HTML, Java, or JavaScript, depending on the type of document. We discuss the use of these interpreters based on the document type later in the chapter (see Figure 4.37).

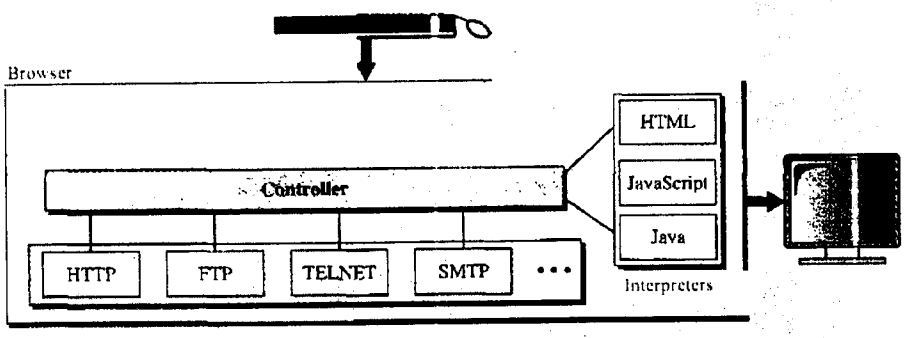


Figure 4.37 Browser

Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested tiles in a cache in memory; memory is faster to access than 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.

Uniform Resource Locator

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying

any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path (see Figure 4.38).

Figure 4.38 URL

The protocol is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the port is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the path name of the tile where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

Cookies

The World Wide Web was originally designed as a stateless entity. A client sends a request; a server responds. Their relationship is over. The original design of WWW, retrieving publicly available documents, exactly fits this purpose. Today the Web has other functions; some are listed here.

I. Some websites need to allow access to registered clients only.

2. Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.

3. Some websites are used as portals: the user selects the Web pages he wants to see.

any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path (see Figure 4.38).

Figure 4.38 URL

The protocol is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the port is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the path name of the tile where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

Cookies

The World Wide Web was originally designed as a stateless entity. A client sends a request; a server responds. Their relationship is over. The original design of WWW, retrieving publicly available documents, exactly fits this purpose. Today the Web has other functions; some are listed here.

I. Some websites need to allow access to registered clients only.

2. Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.

3. Some websites are used as portals: the user selects the Web pages he wants to see.

on the icon of an advertised corporation, a request is sent to the advertising agency. The advertising agency sends the banner, a GIF tile, for example, but it also includes a cookie with the ill of the user. Any future use of the banners adds to the database that profiles the Web behavior of the user. The advertising agency has compiled the interests of the user and can sell this information to other parties. This use of cookies has made them very controversial. Hopefully, some new regulations will be devised to preserve the privacy of users.

4.19 Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) is a language for creating Web pages. The term markup language comes from the book publishing industry. Before a book is typeset and printed, a copy editor reads the manuscript and puts marks on it. These marks tell the compositor how to format the text. For example, if the copy editor wants part of a line to be printed in boldface, he or she draws a wavy line under that part. In the same way, data for a Web page are formatted for interpretation by a browser.

Let us clarify the idea with an example. To make part of a text displayed in boldface with HTML, we put beginning and ending boldface tags (marks) in the text, as shown in Figure 4.39.

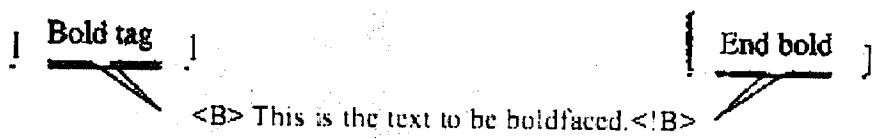


Figure 4.39 Boldface tags

The two tags <B> and </B> are instructions for the browser. When the browser sees these two marks, it knows that the text must be boldfaced (see Figure 4.40).

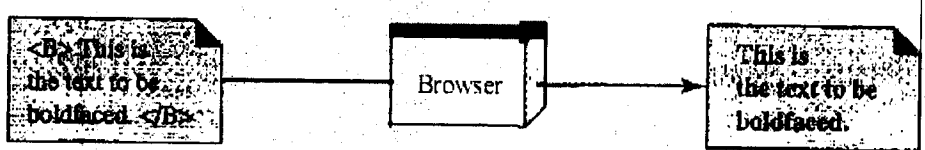


Figure 4.40 Effect of bold face tags

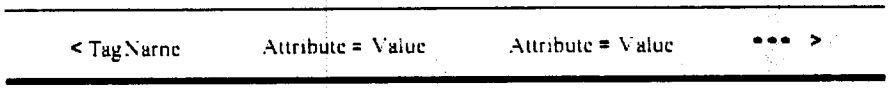
A markup language such as HTML allows us to embed formatting instructions in the tile itself. The instructions are included with the text. In this way, any browser can read the instructions and format the text according to the specific workstation. One might ask why we do not use the

formatting capabilities of word processors to create and save formatted text. The answer is that different word processors use different techniques or procedures for formatting text. For example, imagine that a user creates formatted text on a Macintosh computer and stores it in a Web page. Another user who is on an IBM computer would not be able to receive the Web page because the two computers use different formatting procedures.

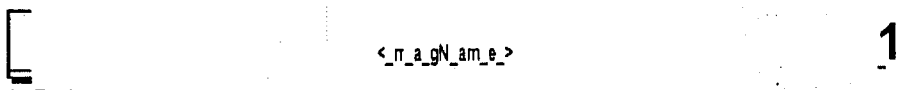
HTML lets us use only ASCII characters for both the main text and formatting instructions. In this way, every computer can receive the whole document as an ASCII document. The main text is the data, and the formatting instructions can be used by the browser to format the data.

A Web page is made up of two parts: the head and the body. The head is the first part of a Web page. The head contains the title of the page and other parameters that the browser will use. The actual contents of a page are in the body, which includes the text and the tags. Whereas the text is the actual information contained in a x1ge, the tags define the appearance of the document. Every HTML tag is a name followed by an optional list of attributes, all enclosed between less-than and greater-than symbols << and >).

An attribute, if present, is followed by an equal's sign and the value of the attribute. Some tags can be used alone; others must be used in pairs. Those that are used in pairs are called beginning and ending tags. The beginning tag can have attributes and values and starts with the name of the g. The ending tag cannot have attributes or values but must have a slash before the name of the tag. The browser makes a decision about the structure of the text based on the tags, which are embedded into the text. Figure 4.41 shows the format of a tag.



a. Beginning tag



b. Ending tag

Figure 4.42 Beginning and ending tags

One commonly used tag category is the text formatting tags such as <B> and <!B>, which make the text bold; <I> and <II>, which make the text italic; and <U> and IV>, which underline the text.

Another interesting tag category is the image tag. Non textual information such as digitized photos or graphic images is not a physical part of an HTML document. But we can use an image tag to point to the tile of a photo or image. The image tag defines the address (URL) of the image to be retrieved. It also specifies how the image can be inserted after retrieval We can choose from several attributes. The most common are SRC (source), which defines the source (address), arid ALIGN, which fines the alignment of the image. The SRC attribute is required. Most browsers accept images in the GIF or IPEG formats. For example, the following tag can retrieve an image stored as imagel.gif in the directory bin/images:

<IMG SRC="/bin/images/image1.gif" ALIGN=MIDDLE >

A third interesting category is the hyperlink g. which is needed to link documents together. Any item (word, phrase, paragraph, or image) can refer to another document through a mechanism called an anchor. The anchor is defined by <A > and <!A> tags, and the anchored item uses the URL to refer to another document. When the document is displayed, the anchored item is underlined, blinking, or boldfaced. The user can click on the anchored item to go to another document, which may or may not be stored on the same server as the original document. The reference phrase is embedded between the beginning and ending tags. The beginning tag can have several attributes, but the one required is HREF hyper1ink reference), which defines the address (URL) of the linked document. For example, the link to the author of a book can be

<A HREF= "http://www.deanza.edu/forouzan"> Author </A>

What appears in the text is the word Author, on which the user can click to go to the author's Web page.

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 output 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 can 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. Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the date program in UNIX and send the result of the pro grain to the client.

4.19 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to cess data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. It is similar to FtP because it transfers files and uses the services of TCP. However, it is much simpler than FtP because it uses only one TCP connection. There is no separate control connection; only data are transferred between the client and the server.

HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP Transaction

Figure 4.43 illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.

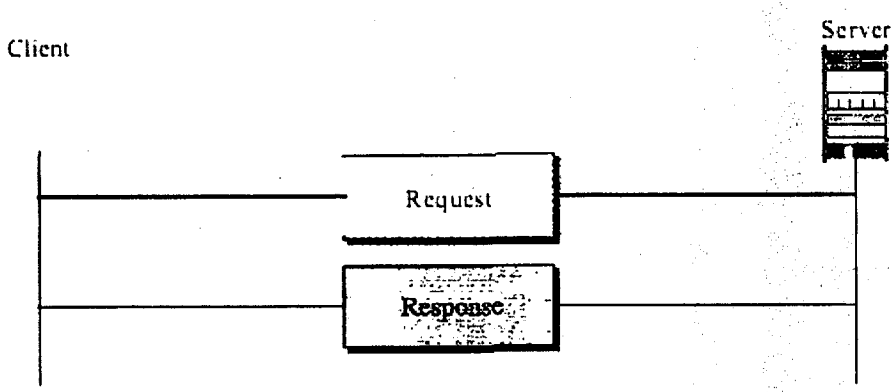


Figure 4.43 HTTP transaction

Messages

The formats of the request and response messages are similar; both are shown in Figure 4.44. A request message consists of a request line, a

header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body.

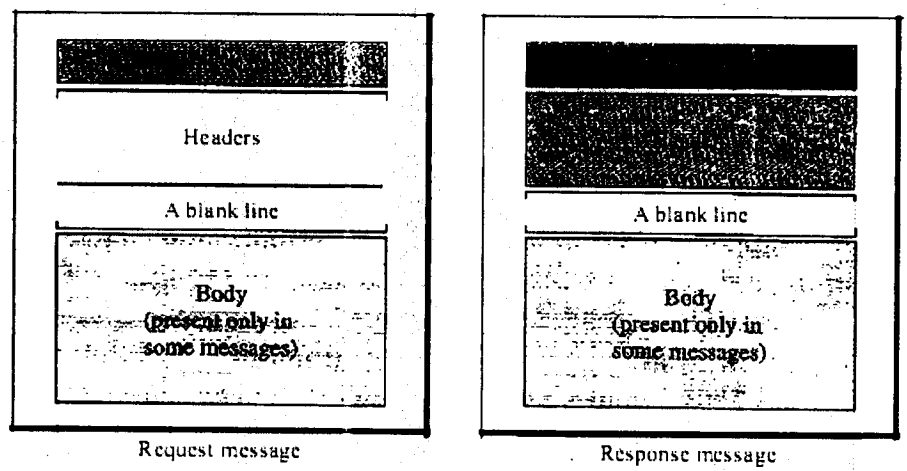


Figure 4.44 Request and response messages

Request and Status Lines The first line in a request message is called a request line; the first line in the response message is called the status line. There is one common field, as shown in Figure 4.45.

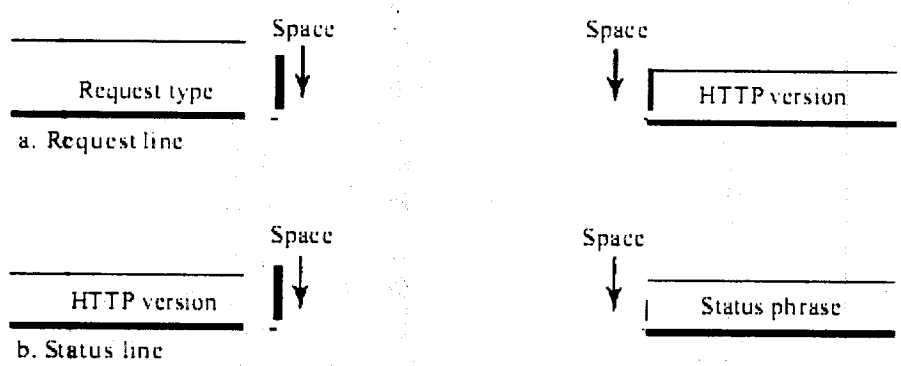


Figure 4.45 Request and status lines

- Request type. This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The request type is categorized into methods as defined in Table 4. 1.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

Table 4.1 Methods

- URL. We discussed the URL earlier in the chapter. Version. The most current version of HTTP is 1.1.

- Status code. This field is used in the response message. The status code field is similar to those in the FTP and the SMTP protocols. 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. We list the most common codes in Table 4.2.

- Status phrase. This field is used in the response message. It explains the status code in text form. Table 4.2 also gives the status phrase.

Code	Phrase	Description
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Table 4.2 Status codes

Header The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value (see Figure 4.46). We will show some header lines in the examples at the end of this chapter. A header

line belongs to one of four categories: general header, request header, response header, and entity header. A request message can contain only general, request, and entity headers. A response message, on the other hand, can contain only general, response, and entity headers.

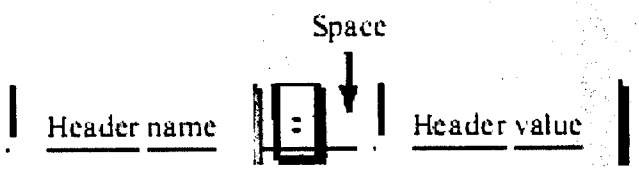


Figure 4.46 Header format

4.20 RPC and Middleware

Most of the services running on an IP network employ the client-server architecture. This involves some actions on the client's side and some actions on the server's side. For example, the client may collect information from the user, process it and send it to the server. The server checks this information for client parameters and upon verifying the parameters processes the request as desire; such client-server applications perform some common actions that are the same for every application. For example, in order to transfer data to a server, the client must request the server and establish an open connection. The applications, which use TCP as part of the data transfer, use the same code lo do so. Hence, it is holier to have the applications to share a common code for performing common tasks in order to reduce the programming effort and errors. Such a software development procedure that relies on the common shared libraries of codes is referred lo as (lie Remote Procedure Call (RPC). The RPC defines small codes to perform specific tasks and the parameters are used to interact with the procedure v The (.Sciails related to a procedure are specified in the parameters (such as the minimum :ind maximum values) and the developers use arguments to pass values into (lie parameters.

Middleware, on the other hand, is similar to RPC and is the term used for software development tools which provide a collection of procedures and interfaces for developing the client-server applications. The collection of shared procedures (objects), bundled as separate pieces is called object-oriented programming. The objects contain methods and well-defined data items which control the behavior of the object.

Some examples of RPC and Middleware are as follows:

- Open Network Computing RPC by Sun Microsystems
- Distribute Computing Environment (DCE) by Open Software Foundation

- Microsoft RPC (MSRPC)
- Component Object Model (COM) by Microsoft
- Common Object Request Broker Architecture (CORBA)

Summary

The main theme of this unit was how to build big networks by interconnecting smaller networks. We looked at bridging in the last unit, but it is a technique that is mostly used to interconnect a small to moderate number of similar networks. What bridging does not do well is tackle the two closely related problems of building very large networks: heterogeneity and scale. The Internet Protocol is the key tool for dealing with these problems, and it provided most of the examples for this unit.

A crucial aspect of the operation of an internetwork is the determination of efficient routes to any destination in the internet. Internet routing algorithms solve this problem in a distributed fashion; this unit introduced the two major classes of algorithms link-state and distance-vector along with examples of their applications.

Naming in the Internet uses a hierarchical scheme called the domain name system (DNS). At the top level are the well-known generic domains, including com and edu as well as about 200 country domains. DNS is implemented as a distributed database system with servers all over the world. DNS holds records with IP addresses, mail exchanges, and other information. By querying a DNS server, a process can map an Internet domain name onto the IP address used to communicate with that domain.

E-mail is one of the two killer apps for the Internet. Everyone from small children to grandparents now use it. Most e-mail systems in the world use the mail system now defined in RFCs 2821 and 2822. Messages sent in this system use system ASCII headers to define message properties. Many kinds of content can be sent using MIME. Messages are sent using SMTP, which works by making a TCP connection from the source host to the destination host and directly delivering the e-mail over the TCP connection.

The other killer app for the Internet is the World Wide Web. The Web is a system for linking hypertext documents. Originally, each document was a page written in HTML with hyperlinks to other documents. Nowadays, XML is gradually starting to take over from HTML.

Review Questions

1. What is routing? What are various activities performed by a router?
2. Differentiate between adaptive and non-adaptive routing.
3. What is a spanning tree?
4. What are the problems with distance vector routing algorithm?
5. What is LSP?
6. Explain Hierarchical Routing algorithm
7. Describe Multicast Routing algorithm
8. Describe the functions of the two FTP connections.
9. What kinds of tile types can FTP transfer?
10. Describe the addressing system used by SMTP.
11. How is HTTP related to WWW?
12. How is HTTP similar to SMTP?
13. How is HTTP similar to FTP?
14. Define network management.
15. List five functions of network management.
16. Define configuration management and its purpose.

Reference Books

1. Computer Networking: Schaum's outlines (TMH).
2. Kurose J F & Ross K.W: Computer Networking (Pearson)
3. Tanenbaum A S: Computer Networks (PHI) 4th Ed.
4. Data Communication & Networking – Behrouz Forouzan(TMH)

UNIT – V

Structure

5. 1 Introduction

The word cryptography has come from a Greek word, which means secret writing. In the present day context it refers to the tools and techniques used to make messages secure for communication between the participants and make messages immune to attacks by hackers. For private communication through public network, cryptography plays a very crucial role. The role of cryptography can be illustrated with the help a simple model of cryptography as shown in Figure 5.1. The message to be sent through an unreliable medium is known as plaintext, which is encrypted before sending over the medium. The encrypted message is known as cipher text, which is received at the other end of the medium and decrypted to get back the original plaintext message. In this lesson we shall discuss various cryptography algorithms, which can be divided into two broad categorize

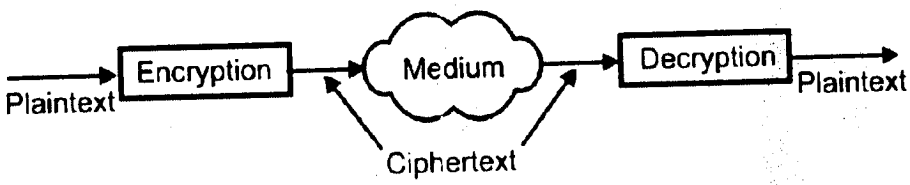


Figure 5.1 A simple cryptography model

5.2 Symmetric Key Cryptography

The cipher, an algorithm that is used for converting the plaintext to ciphertext operates on a key, which is essentially a specially generated number (value). To decrypt a secret message (ciphertext) to get back the original message (plaintext), a decrypt algorithm uses a decrypt key. In symmetric key cryptography, same key is shared, i.e. the same key is used in both encryption and decryption as shown in Figure 5.2. The algorithm used to decrypt is just the inverse of the algorithm used for encryption. For example, if addition and division is used for encryption, multiplication and subtraction are to be used for decryption.

Symmetric key cryptography algorithms are simple requiring lesser execution time. As a consequence, these are commonly used for long messages. However, these algorithms suffer from the following limitations:

- Requirement of large number of unique keys. For example for n users the number of keys required is n (n- 1)/2.
- Distribution of keys among the users in a secured manner is difficult.

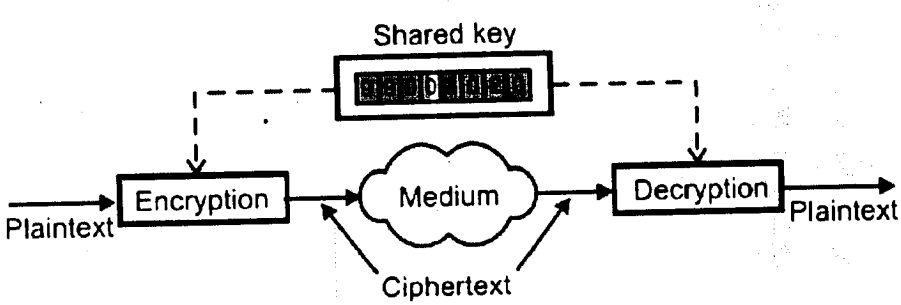


Figure 5.2. A simple symmetric key cryptography model

Monoalphabetic Substitution

One simple example of symmetric key cryptography is the Monoalphahetic substitution. In this case, the relationship between a character in the plaintext and a character in the ciphertext is always one-to-one. An example Monoalphabetic substitution is the Caesar cipher. As shown in Figure 5.3, in this approach a character in the ciphertext is substituted by another character shifted by three places, e.g. A is substituted by D. Key feature of this approach is that it is very simple but the code can be attacked very easily.

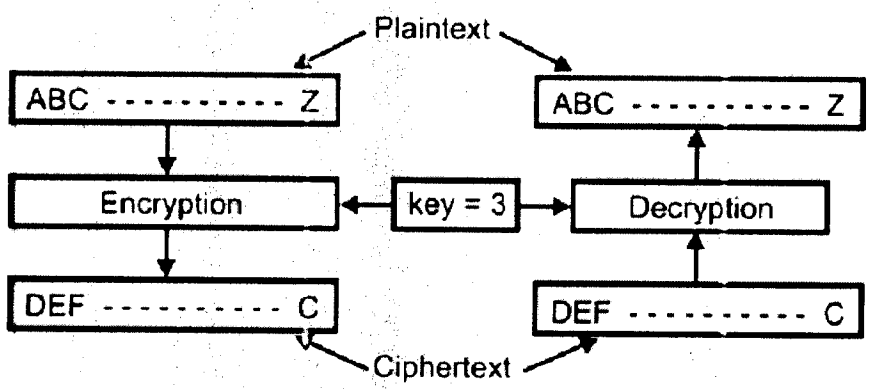


Figure 5.3. The Caesar cipher

Polyalphabetic Substitution

This is an improvement over the Caesar cipher. Here the relationship between a character in the plaintext and a character in the ciphertext is always one-to-many.

Example 5: Example of polyalphabetic substitution is the Vigenere cipher. In this case, a particular character is substituted by different characters in the ciphertext depending on its position in the plaintext. Figure 5.4 explains the polyalphabetic substitution. Here the top row shows different characters in the plaintext and the characters in different bottom rows show the characters by which a particular character is to be replaced depending upon its position in different rows from row-0 to row-25.

• Key feature of this approach is that it is more complex and the code is harder to attack successfully.

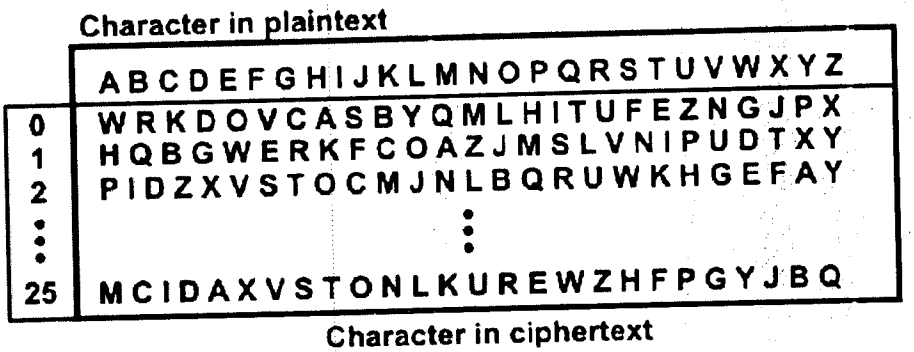
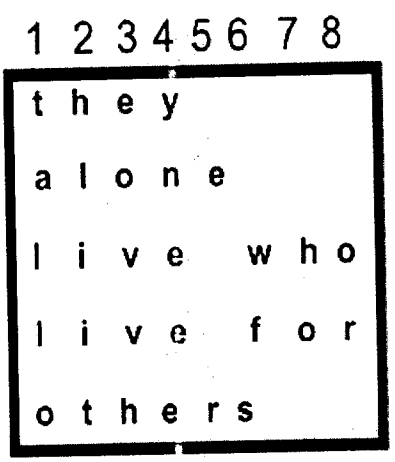


Figure 5.4. Polyalphabetric substitution

Transpositional Cipher

The transpositional cipher, the characters remain unchanged but their positions are changed to create the ciphertext. Figure 5.5 illustrates how five lines of a text get modified using transpositional cipher. The characters are arranged in two-dimensional matrix and columns are interchanged according to a key is shown in the middle portion of the diagram. The key defines which columns are to be swapped. As per the key shown in the figure, character of column is to be swapped to column 3, character of column 2 is to be swapped to column 6, and so on. Decryption can be done by swapping in the reverse order using the same key.

Transpositional cipher is also not a very secure approach. The attacker can find the plaintext by trial and error utilizing the idea of the frequency of occurrence of characters.



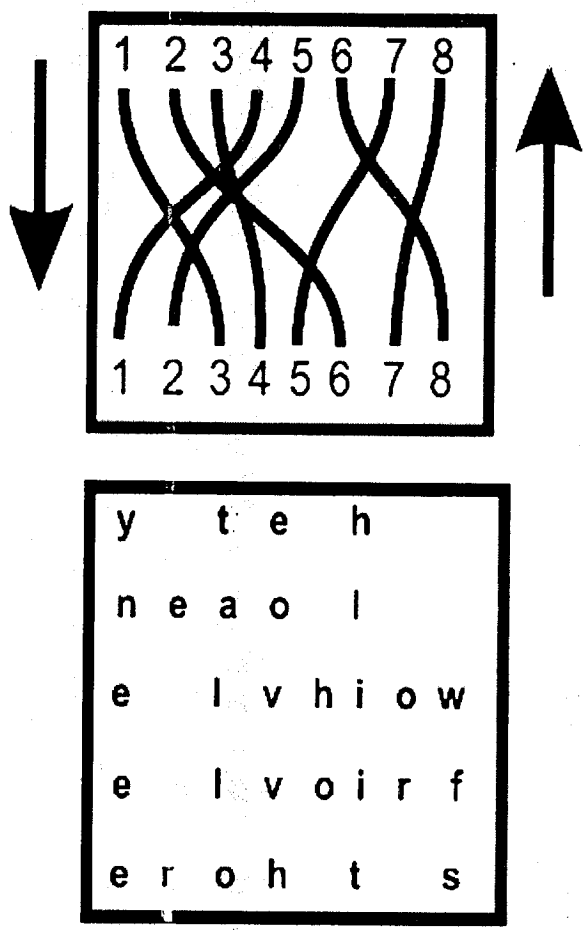


Figure 5.5. Operation of a transpositional cipher

Block Ciphers

Block ciphers use a block of bits as the unit of encryption and decryption. To encrypt a 64-bit block, one has to take each of the 264 input values and map it to one of the 264 output values. The mapping should be one-to-one. Encryption and decryption operations of a block cipher are shown in Figure 8. 1.6. Some operations, such as permutation and substitution, are performed on the block of bits based on a key (a secret number) to produce another block of bits. The permutation and substitution operations are shown in Figs 5.7 and 5.8, respectively. In the decryption process, operations are performed in the reverse order based on the same key to get back the original block of bits.

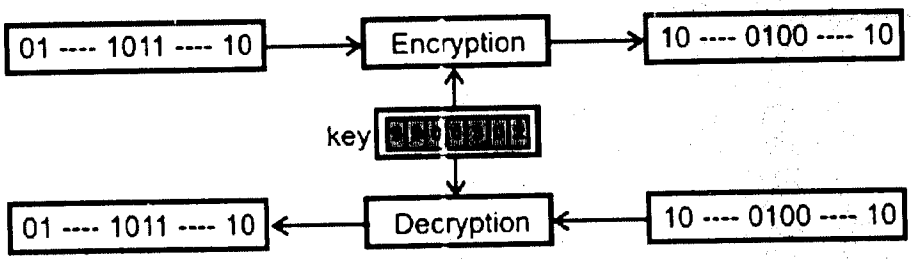


Figure 5.6. Transformations in Block Ciphers

Permutation: As shown in Figure 5.7, the permutation is performed by a permutation box at the bit-level, which keeps the number of Os and is same at the input and output. Although it can be implemented either by hardware or software, the hardware implementation is faster.

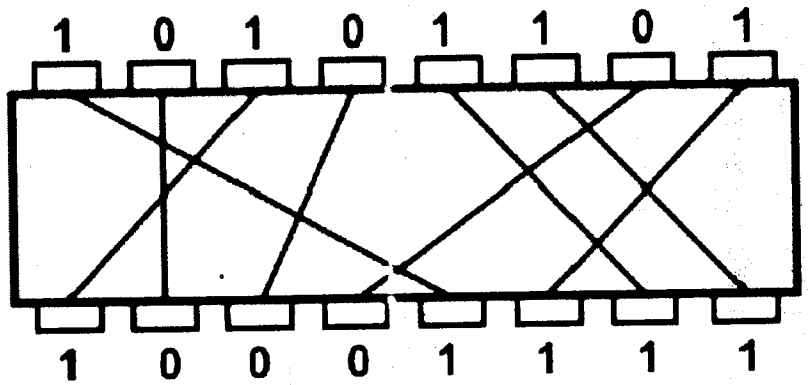


Figure 5.7 Permutation operation used in Block Ciphers

Substitution: As shown in Figure 5.8, the substitution is implemented with the help of three building blocks a decoder, one p-box and an encoder. For an n-bit input, the decoder produces a 2 bit output having only one 1, which is applied to the P-box. The P-box permutes the output of the decoder and it is applied to the encoder. The encoder, in turn, produces an n-bit output. For example, if the input to the decoder is 0 11, the output of the decoder is 00001000. Let the permuted output is 01000000, the output of the encoder is 0 11.

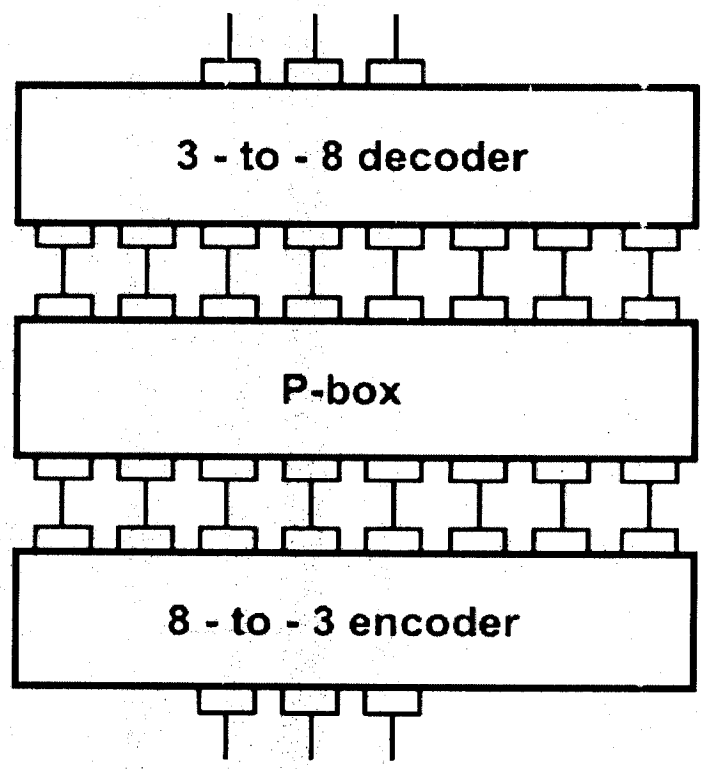


Figure 5.8. Substitution operation used in Block Ciphers

A block Cipher: A block cipher realized by using substitution and permutation operations is shown in Figure 5.9. It performs the following steps:

Step-1: Divide input into 8-bit pieces

Step-2: Substitute each 8-bit based on functions derived from the key

Step-3: Permute the bits based on the key

All the above three steps are repeated for an optimal number of rounds.

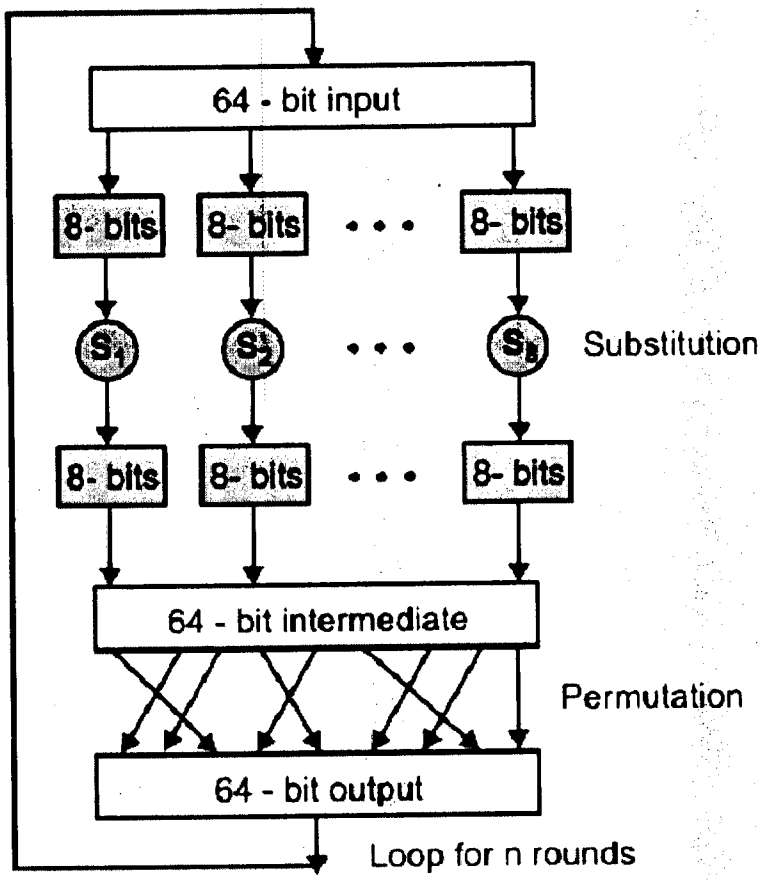


Figure 5.9.Encryption by using substitution and permutation

Data Encryption Standard (DES)

One example of the block cipher is the Data Encryption Standard (DES). Basic features of the DES algorithm are given below:

- A monoalphabetic substitution cipher using a 64-bit character
- It has 1 9 distinct stages
- Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length.
- The decryption can be done with the same password; the stages must then be carried out in reverse order.
- DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the ciphertext.

• As the number of rounds increases, the security of the algorithm increases exponentially.

• Once the key scheduling and plaintext preparation have been completed, the actual encryption or decryption is performed with the help of the main DES algorithm as shown in Figure 5.10.

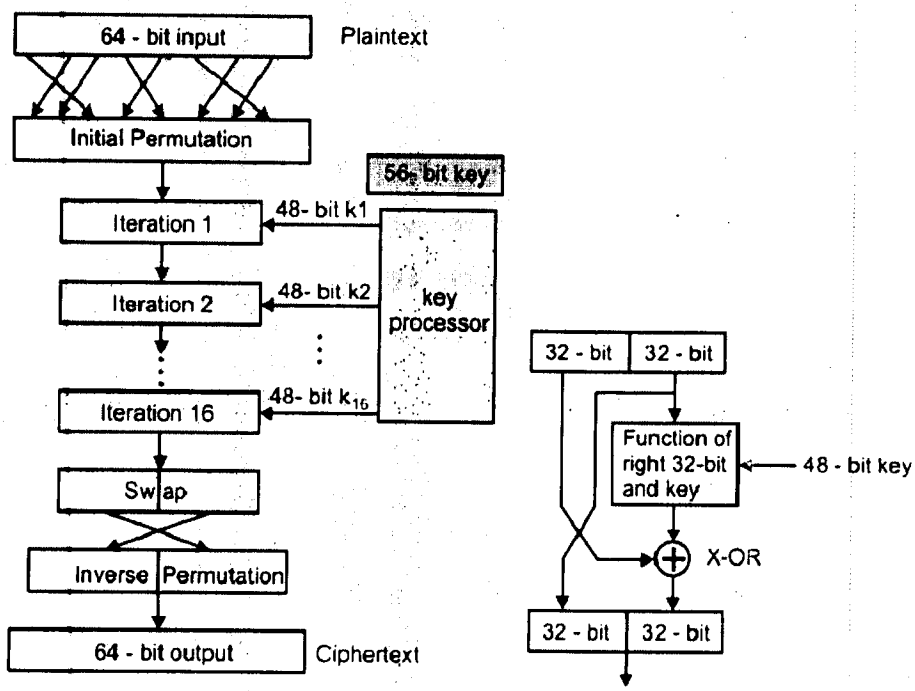


Figure 5.10 64-bit Data Encryption Standard (DES)

Encrypting a Large Message

DES can encrypt a block of 64 bits. However, to encrypt blocks of larger size, there exist several modes of operation as follows:

- Electronic Code Book (ECB)
- Cipher Block Chaining (C BC')
- Cipher Feedback Mode (CFB)
- Output Feedback Mode (OFB)

Electronic Code Book (ECB)

This is part of the regular DES algorithm. Data is divided into 64-bit blocks and each block is encrypted one at a time separately as shown in Figure 5.

11. Separate encryptions ith different blocks are totally independent of each other.

Disadvantages of ECB

- If a message contains two identical blocks of 64-bits, the ciphertext corresponding to these blocks are identical. This may give some information to the eavesdropper
- Someone can modify or rearrange blocks to his own advantage
- Because of these flaws, ECB is rarely used

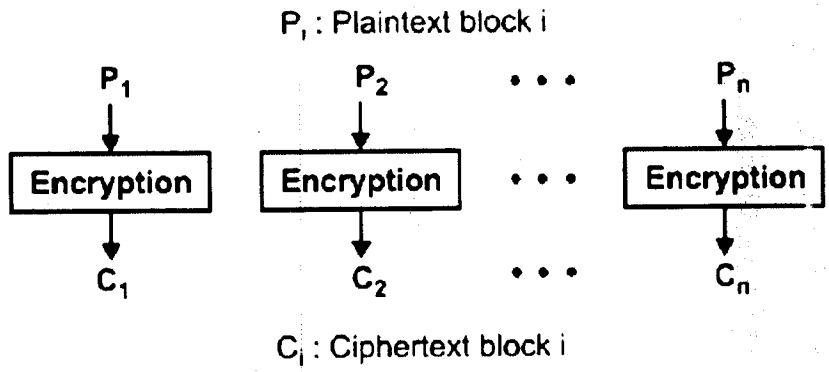


Figure 5.11 Electronic Code Book (ECB) encryption technique

Cipher Block Chaining (C BC)

In this mode of operation, encrypted ciphertext of each block of ECB is XORed with the next plaintext block to be encrypted, thus making all the blocks dependent on all the previous blocks. The initialization vector is sent along with data as shown in Figure 5.12.

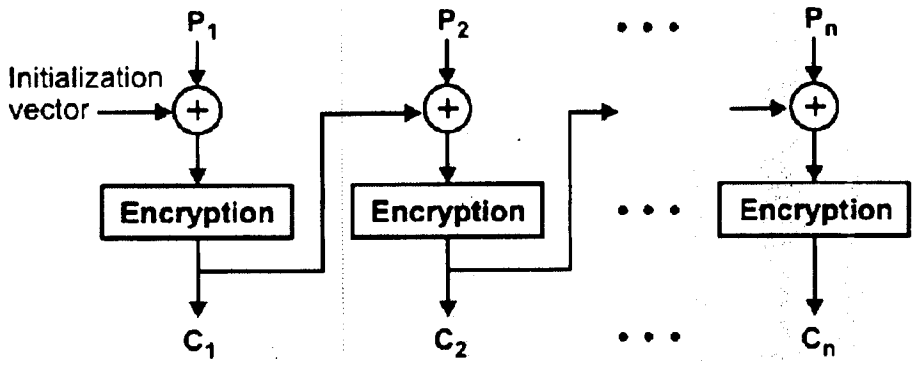


Figure 5.12 Cipher Block Chaining (CBC) encryption technique

Cipher Feedback Mode (CFB)

- In this mode, blocks of plaintext that is less than 64 bits long can be encrypted as shown in Figure 5.13.
- This is commonly used with interactive terminals
- It can receive and send k bits (say k8) at a time in a streamed manner

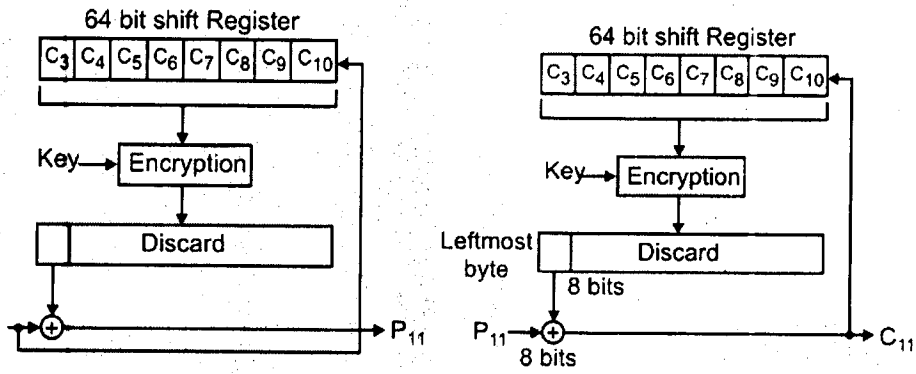


Figure 5.13 Cipher Feedback Mode (CFB) encryption technique

Output Feedback Mode (OFB)

The encryption technique of Output Feedback Mode (OFB) is shown in Figure 5.14. Key features of this mode are mentioned below:

- OFB is also a stream cipher
- Encryption is performed by XORing the message with the one-time pad
- One-time pad can be generated in advance
- If some bits of the ciphertext get garbled, only those bits of plaintext get garbled
- The message can be of any arbitrary size
- Less secure than other modes

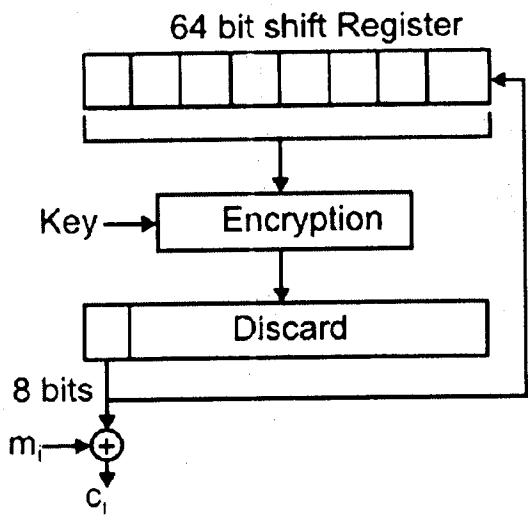


Figure 5.14 Output Feedback Mode (OFB) encryption technique

Triple DES

Triple DES, popularly known as 3 DES, is used to make DES more secure by effectively increasing the key length. Its operation is explained below:

- Each block of plaintext is subjected to encryption by K!, decryption by K2 and again encryption by K! in a sequence as shown in Figure 5.15
- CBC is used to turn the block encryption scheme into a stream encryption scheme

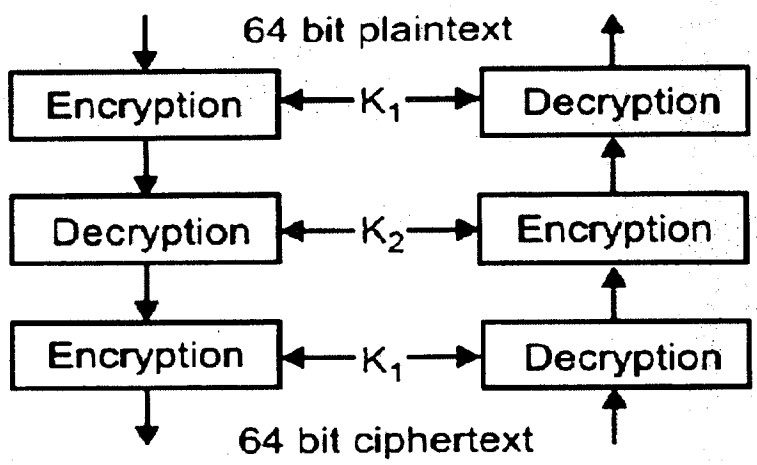


Figure 5.15 Triple DES encryption technique

5.3 Public key Cryptography

In public key cryptography, there are two keys: a private key and a public key. The public key is announced to the public, where as the private key is kept by the receiver. The sender uses the public key of the receiver for encryption and the receiver uses his private key for decryption as shown in Figure 5. 16.

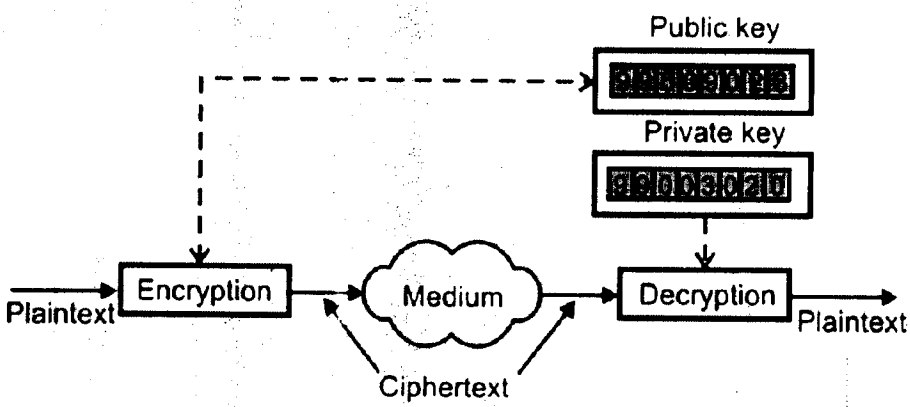


Figure 5.16 Public key encryption technique

• Advantages:

- The pair of keys can be used with any other entity
- The number of keys required is small

• Disadvantages:

- It is not efficient for long messages
- Association between an entity and its public key must be verified

RSA

The most popular public-key algorithm is the RSA (named after their inventors Rivest, Shamir and Adleman) as shown in Figure 5.17. Key features of the RSA algorithm are given below:

- Public key algorithm that performs encryption as well as decryption based on number theory
- Variable key length; long for enhanced security and short for efficiency (typical 512 bytes)
- Variable block size, smaller than the key length

- The private key is a pair of numbers (d, n) and the public key is also a pair of numbers (e, n)
- Choose two large primes p and q (typically around 256 bits)
- Compute n = p x q and z = (p-1)x(q- 1)
- Choose a number d relatively prime to z
- Find e such that e x d mod (p-1)x(q- 1) = 1
- For encryption: C Pe (mod n) For decryption: P = Cd (mod n)

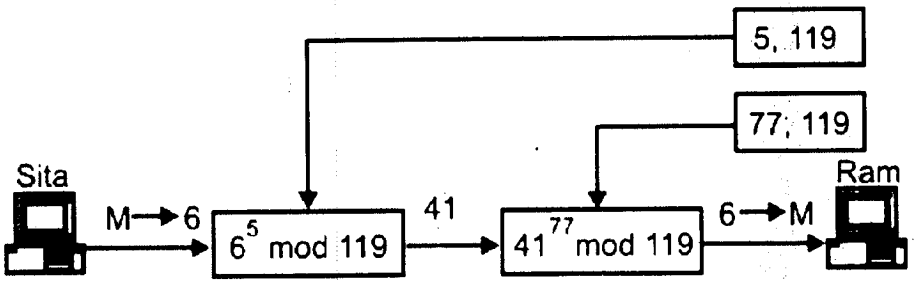


Figure 5.17 The RSA public key encryption technique

Other Public-Key Algorithms

Although RSA is widely used, it is by no means the only public-key algorithm known. The first public-key algorithm was the knapsack algorithm. The idea here is that someone owns a large number of objects, each with a different weight. The owner encodes the message by secretly selecting a subset of the objects and placing them in the knapsack. The total weight of the objects in the knapsack is made public, as is the list of all possible objects. The list of objects in the knapsack is kept secret. With certain additional restrictions, the problem of figuring out a possible list of objects with the given weight was thought to be computationally infeasible and formed the basis of the public-key algorithm.

The algorithm's inventor, Ralph Merkle, was quite sure that this algorithm could not be broken, so he offered a $100 reward to anyone who could break it. Adi Shamir (the "S" in RSA) promptly broke it and collected the reward. Undeterred, Merkle strengthened the algorithm and offered a $1000 reward to anyone who could break the new one. Ronald Rivest (the "R" in RSA) promptly broke the new one and collected the reward. Merkle did not dare offer $10,000 for the next version, so "A" (Leonard Adleman) was out of luck. Nevertheless, the knapsack algorithm is not considered secure and is not used in practice any more.

Other public-key schemes are based on the difficulty of computing discrete logarithms. A few other schemes exist, such as those based on elliptic curves, but the two major categories are those based on the difficulty of factoring large numbers and computing discrete logarithms modulo a large prime. These problems are thought to be genuinely difficult to solve mathematicians have been working on them for many years without any great break through.

5.4 Digital Signatures

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

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

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

The first requirement is needed, for example, in financial systems. When a customer's computer orders a bank's computer to buy a ton of gold, the bank's computer needs to be able to make sure that the computer giving the order really belongs to the company whose account is to be debited. In other words, the bank has to authenticate the customer (and the customer has to authenticate the bank).

The second requirement is needed to protect the bank against fraud. Suppose that the bank buys the ton of gold, and immediately thereafter the price of gold drops sharply. A dishonest customer might sue the bank, claiming that he never issued any order to buy gold. When the bank produces the message in court, the customer denies having sent it. The property that no party to a contract can later deny having signed it is called

nonrepudiation. The digital signature schemes that we will now study help provide it.

The third requirement is needed to protect the customer in the event that the price of gold shoots up and the bank tries to construct a signed message in which the customer asked for one bar of gold instead of one ton. In this fraud scenario, the bank just keeps the rest of the gold for itself.

Symmetric-Key Signatures

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

When Alice wants to send a signed plaintext message, P, to her banker, Bob, she generates $K_A(B, R_A, t, P)$, where B is Bob's identity, R_A is a random number chosen by Alice, t is a timestamp to ensure freshness, and $K_A(B, R_A, t, P)$ is the message encrypted with her key, K_A. Then she sends it as depicted in Figure 5.18. BB sees that the message is from Alice, decrypts it, and sends a message to Bob as shown. The message to Bob contains the plaintext of Alice's message and also the signed message K_{BB} (A, t, P). Bob now carries out Alice's request.

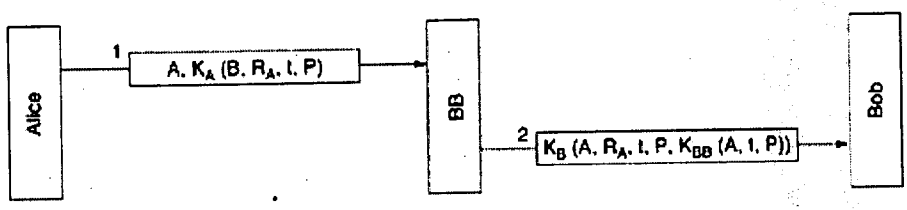


Figure 5.18. Digital signatures with Big Brother.

What happens if Alice later denies sending the message? Step 1 is that everyone sues everyone (at least, in the United States). Finally, when the case comes to court and Alice vigorously denies sending Bob the disputed message, the judge will ask Bob how he can be sure that the disputed message came from Alice and not from Trudy. Bob first points out that BB will not accept a message from Alice unless it is encrypted with K_A, so there is no possibility of Trudy sending BB a false message from Alice without BB detecting it immediately.

Bob then dramatically produces Exhibit A: K_{BB} (A, t, P). Bob says that this is a message signed by BB which proves Alice sent P to Bob. The judge then asks BB (whom everyone trusts) to decrypt Exhibit A. When BB testifies that Bob is telling the truth, the judge decides in favor of Bob. Case dismissed.

One potential problem with the signature protocol of Figure 5.18 is Trudy replaying either message. To minimize this problem, timestamps are used throughout. Furthermore, Bob can check all recent messages to see if R_A was used in any of them. If so, the message is discarded as a replay. Note that based on the timestamp, Bob will reject very old messages. To guard against instant replay attacks, Bob just checks the R_A of every incoming message to see if such a message has been received from Alice in the past hour. If not, Bob can safely assume this is a new request.

Public-Key Signatures

A structural problem with using symmetric-key cryptography for digital signatures is that everyone has to agree to trust Big Brother. Furthermore, Big Brother gets to read all signed messages. The most logical candidates for running the Big Brother server are the government, the banks, the accountants, and the lawyers. Unfortunately, none of these organizations inspire total confidence in all citizens. Hence, it would be nice if signing documents did not require a trusted authority.

Fortunately, public-key cryptography can make an important contribution in this area. Let us assume that the public-key encryption and decryption algorithms have the property that E(D(P)) = P in addition, of course, to the usual property that D(E(P)) = P. (RSA has this property, so the assumption is not unreasonable.) Assuming that this is the case, Alice can send a signed plaintext message, P, to Bob by transmitting $E_B(D_A(P))$. Note carefully that Alice knows her own (private) key, D_A, as well as Bob's public key, E_B, so constructing this message is something Alice can do.

When Bob receives the message, he transforms it using his private key, as usual, yielding $D_A(P)$, as shown in 19 He stores this text in a safe place and then applies E_A to get the original plaintext.

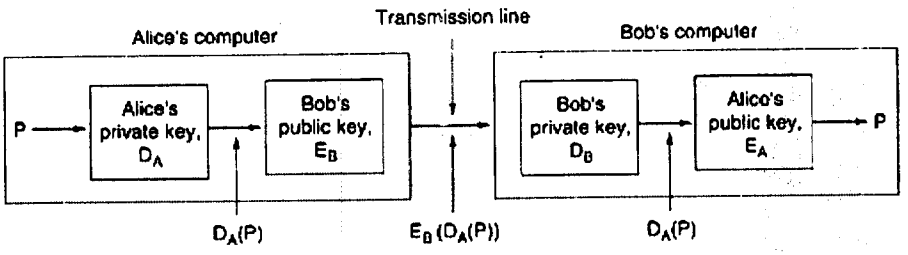


Figure 5.19. Digital signatures using public-key cryptography.

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

Although using public-key cryptography for digital signatures is an elegant scheme, there are problems that are related to the environment in which they operate rather than with the basic algorithm. For one thing, Bob can prove that a message was sent by Alice only as long as D_A remains secret. If Alice discloses her secret key, the argument no longer holds, because anyone could have sent the message, including Bob himself.

The problem might arise, for example, if Bob is Alice's stockbroker. Alice tells Bob to buy a certain stock or bond. Immediately thereafter, the price drops sharply. To repudiate her message to Bob, Alice runs to the police claiming that her home was burglarized and the PC holding her key was stolen. Depending on the laws in her state or country, she may or may not be legally liable, especially if she claims not to have discovered the break-in until getting home from work, several hours later.

Another problem with the signature scheme is what happens if Alice decides to change her key. Doing so is clearly legal, and it is probably a good idea to do so periodically. If a court case later arises, as described above, the judge will apply the current E_A to $D_A(P)$ and discover that it does not produce P. Bob will look pretty stupid at this point.

In principle, any public-key algorithm can be used for digital signatures. The de facto industry standard is the RSA algorithm. Many security products

use it. However, in 1991, NIST proposed using a variant of the El Gamal public-key algorithm for their new Digital Signature Standard (DSS). El Gamal gets its security from the difficulty of computing discrete logarithms, rather than from the difficulty of factoring large numbers.

As usual when the government tries to dictate cryptographic standards, there was an uproar. DSS was criticized for being

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

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

Message Digests

One criticism of signature methods is that they often couple two distinct functions: authentication and secrecy. Often, authentication is needed but secrecy is not. Also, getting an export license is often easier if the system in question provides only authentication but not secrecy. Below we will describe an authentication scheme that does not require encrypting the entire message.

This scheme is based on the idea of a one-way hash function that takes an arbitrarily long piece of plaintext and from it computes a fixed-length bit string. This hash function, MD, often called a message digest, has four important properties:

1. Given P, it is easy to compute MD(P).
2. Given MD(P), it is effectively impossible to find P.
3. Given P no one can find P' such that MD (P') = MD(P).
4. A change to the input of even 1 bit produces a very different output.

To meet criterion 3, the hash should be at least 128 bits long, preferably more. To meet criterion 4, the hash must mangle the bits very thoroughly, not unlike the symmetric-key encryption algorithms we have seen.

Computing a message digest from a piece of plaintext is much faster than encrypting that plaintext with a public-key algorithm, so message digests can be used to speed up digital signature algorithms. To see how this works, consider the signature protocol of Figure 5.18 again. Instead of signing P with K_{BB} (A, t, P), BB now computes the message digest by applying MD to P, yielding MD(P). BB then encloses K_{BB} (A, t, MD(P)) as the fifth item in the list encrypted with K_B that is sent to Bob, instead of K_{BB} (A, t, P).

If a dispute arises, Bob can produce both P and K_{BB} (A, t, MD(P)). After Big Brother has decrypted it for the judge, Bob has MD(P), which is guaranteed to be genuine, and the alleged P. However, since it is effectively impossible for Bob to find any other message that gives this hash, the judge will easily be convinced that Bob is telling the truth. Using message digests in this way saves both encryption time and message transport costs.

Message digests work in public-key cryptosystems, too, as shown in Figure 5.20 Here, Alice first computes the message digest of her plaintext. She then signs the message digest and sends both the signed digest and the plaintext to Bob. If Trudy replaces P underway, Bob will see this when he computes MD(P) himself.

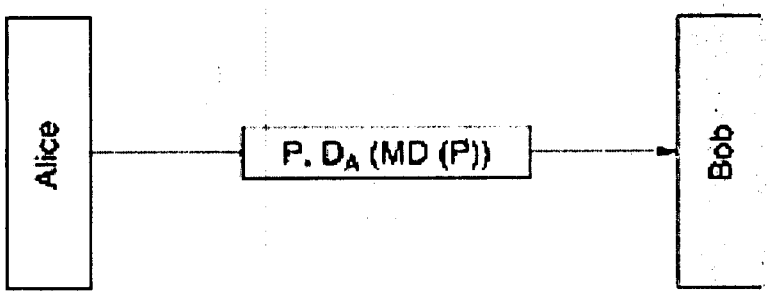


Figure 5.20. Digital signatures using message digests.

MD5

A variety of message digest functions have been proposed. The most widely used ones are MD5 and SHA-1. MD5 is the fifth in a series of message digests designed by Ronald Rivest. It operates by mangling bits

in a sufficiently complicated way that every output bit is affected by every input bit. Very briefly, it starts out by padding the message to a length of 448 bits (modulo 512). Then the original length of the message is appended as a 64-bit integer to give a total input whose length is a multiple of 512 bits. The last precomputation step is initializing a 128-bit buffer to a fixed value.

Now the computation starts. Each round takes a 512-bit block of input and mixes it thoroughly with the 128-bit buffer. For good measure, a table constructed from the sine function is also thrown in. The point of using a known function like the sine is not because it is more random than a random number generator, but to avoid any suspicion that the designer built in a clever back door through which only he can enter. Remember that IBM's refusal to disclose the principles behind the design of the S-boxes in DES led to a great deal of speculation about back doors. Rivest wanted to avoid this suspicion. Four rounds are performed per input block. This process continues until all the input blocks have been consumed. The contents of the 128-bit buffer form the message digest.

MD5 has been around for over a decade now, and many people have attacked it. Some vulnerabilities have been found, but certain internal steps prevent it from being broken. However, if the remaining barriers within MD5 fall, it may eventually fail. Nevertheless, at the time of this writing, it was still standing.

SHA-1

The other major message digest function is SHA-1 (Secure Hash Algorithm 1), developed by NSA and blessed by NIST in FIPS 180-1. Like MD5, SHA-1 processes input data in 512-bit blocks, only unlike MD5, it generates a 160-bit message digest. A typical way for Alice to send a nonsecret but signed message to Bob is illustrated in Figure 5.21. Here her plaintext message is fed into the SHA-1 algorithm to get a 160-bit SHA-1 hash. Alice then signs the hash with her RSA private key and sends both the plaintext message and the signed hash to Bob.

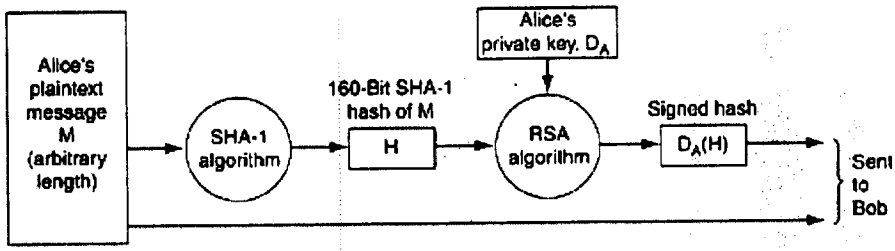


Figure 5.21. Use of SHA-1 and RSA for signing nonsecret messages.

After receving the message, Bob computes the SHA-1 hash himself and also applies Alice's public key to the signed hash to get the original hash, H. If the two agree, the message is considered valid. Since there is no way for Trudy to modify the (plaintext) message while its is in transit and produce a new one that hashes to H, Bob can easily detect any changes Trudy has made to the message. For messages whose integrity is important but whose contents are not secret, the scheme of Figure 5.21 is widely used. For a relatively small cost in computation, it guarantees that any modifications made to the plaintext message in transit can be detected with very high probability.

Now let us briefly see how SHA-1 works. It starts out by padding the message by adding a 1 bit to the end, followed by as many 0 bits as are needed to make the length a multiple of 512 bits. Then a 64-bit number containing the message length before padding is ORed into the low-order 64 bits. In Figure 5.22, the message is shown with padding on the right because English text and figures go from left to right (i.e., the lower right is generally perceived as the end of the figure). With computers, this orientation corresponds to big-endian machines such as the SPARC, but SHA-1 always pads the end of the message, no matter which endian machine is used.

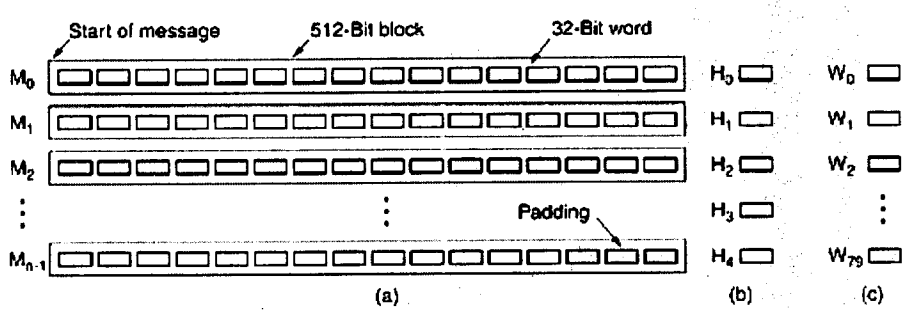


Figure 8-22. (a) A message padded out to a multiple of 512 bits. (b) The output variables. (c) The word array.

During the computation, SHA-1 maintains five 32-bit variables, H_0 through H_4, where the hash accumulates. These are shown in Figure 8-22(b). They are initialized to constants specified in the standard.

Each of the blocks M_0 through M_{n-1} is now processed in turn. For the current block, the 16 words are first copied into the start of an auxiliary 80-word array, W, as shown in Figure 8-22(c) Then the other 64 words in W are filled in using the formula

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

where $S^b(W)$ represents the left circular rotation of the 32-bit word, W, by b bits. Now five scratch variables, A through E are initialized from H_0 through H_4, respectively.

The actual calculation can be expressed in pseudo-C as

```
for (i = 0; i < 80; i++)
{
  temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
  E=D; D=C; C=S^30(B); B = A; A = temp;
}
```

where the K_i constants are defined in the standard. The mixing functions f_i are defined as

$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D)$ $(0 \le i \le 19)$
$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D$ $(20 \le i \le 39)$
$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D)$ $(40 \le i \le 59)$
$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D$ $(60 \le i \le 79)$

When all 80 iterations of the loop are completed, A through E are added to H_0 through H_4, respectively.

Now that the first 512-bit block has been processed, the next one is started. The W array is reinitialized from the new block, but H is left as it was. When this block is finished, the next one is started, and so on, until all the 512-bit message blocks have been tossed into the soup. When the last block has

been finished, the five 32-bit words in the H array are output as the 160-bit cryptographic hash. The complete C code for SHA-1 is given in RFC 3174.

New versions of SHA-1 are under development for hashes of 256, 384, and 512 bits, respectively.

The Birthday Attack

In the world of crypto, nothing is ever what it seems to be. One might think that it would take on the order of 2^m operations to subvert an m-bit message digest. In fact, $2^{m/2}$ operations will often do using the birthday attack, an approach published by Yuval in his now-classic paper "How to Swindle Rabin."

The idea for this attack comes from a technique that math professors often use in their probability courses. The question is: How many students do you need in a class before the probability of having two people with the same birthday exceeds 1/2? Most students expect the answer to be way over 100. In fact, probability theory says it is just 23. Without giving a rigorous analysis, intuitively, with 23 people, we can form (23 x 22)/2 = 253 different pairs, each of which has a probability of 1/365 of being a hit. In this light, it is not really so surprising any more.

More generally, if there is some mapping between inputs and outputs with n inputs (people, messages, etc.) and k possible outputs (birthdays, message digests, etc.), there are n(n - 1)/2 input pairs. If n(n - 1)/2 >k, the chance of having at least one match is pretty good. Thus, approximately, a match is likely for $n > \sqrt{k}$. This result means that a 64-bit message digest can probably be broken by generating about 2^{32} messages and looking for two with the same message digest.

Let us look at a practical example. The Department of Computer Science at State University has one position for a tenured faculty member and two candidates, Tom and Dick. Tom was hired two years before Dick, so he goes up for review first. If he gets it, Dick is out of luck. Tom knows that the department chairperson, Marilyn, thinks highly of his work, so he asks her to write him a letter of recommendation to the Dean, who will decide on Tom's case. Once sent, all letters become confidential.

Marilyn tells her secretary, Ellen, to write the Dean a letter, outlining what she wants in it. When it is ready, Marilyn will review it, compute and sign

the 64-bit digest, and send it to the Dean. Ellen can send the letter later by e-mail.

Unfortunately for Tom, Ellen is romantically involved with Dick and would like to do Tom in, so she writes the letter below with the 32 bracketed options.

5.5 Management of Public Keys

Public-key cryptography makes it possible for people who do not share a common key to communicate securely. It also makes signing messages possible without the presence of a trusted third party. Finally, signed message digests make it possible to verify the integrity of received messages easily.

However, there is one problem that we have glossed over a bit too quickly: if Alice and Bob do not know each other, how do they get each other's public keys to start the communication process? The obvious solution put your public key on your Web site does not work for the following reason. Suppose that Alice wants to look up Bob's public key on his Web site. How does she do it? She starts by typing in Bob's URL. Her browser then looks up the DNS address of Bob's home page and sends it a GET request, as shown in Figure 5.23. Unfortunately, Trudy intercepts the request and replies with a fake home page, probably a copy of Bob's home page except for the replacement of Bob's public key with Trudy's public key. When Alice now encrypts her first message with E_T, Trudy decrypts it, reads it, reencrypts it with Bob's public key, and sends it to Bob, who is none the wiser that Trudy is reading his incoming messages. Worse yet, Trudy could modify the messages before reencrypting them for Bob. Clearly, some mechanism is needed to make sure that public keys can be exchanged securely.

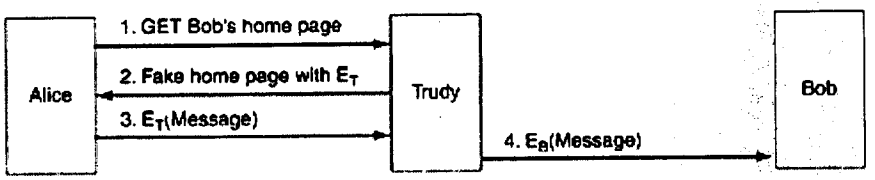


Figure 5.23. A way for Trudy to subvert public-key encryption.

Certificates

As a first attempt at distributing public keys securely, we could imagine a key distribution center available on-line 24 hours a day to provide public keys on demand. One of the many problems with this solution is that it is not scalable, and the key distribution center would rapidly become a bottleneck. Also, if it ever went down, Internet security would suddenly grind to a halt.

For these reasons, people have developed a different solution, one that does not require the key distribution center to be on-line all the time. In fact, it does not have to be on-line at all. Instead, what it does is certify the public keys belonging to people, companies, and other organizations. An organization that certifies public keys is now called a CA (Certification Authority).

As an example, suppose that Bob wants to allow Alice and other people to communicate with him securely. He can go to the CA with his public key along with his passport or driver's license and ask to be certified. The CA then issues a certificate similar to the one in Figure 5.24 and signs its SHA-1 hash with the CA's private key. Bob then pays the CA's fee and gets a floppy disk containing the certificate and its signed hash.

I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA's private key

Figure 5.24. A possible certificate and its signed hash.

The fundamental job of a certificate is to bind a public key to the name of a principal (individual, company, etc.). Certificates themselves are not secret or protected. Bob might, for example, decide to put his new certificate on his Web site, with a link on the main page saying: Click here for my public-

key certificate. The resulting click would return both the certificate and the signature block (the signed SHA-1 hash of the certificate).

Now let us run through the scenario of Figure 5.23 again. When Trudy intercepts Alice's request for Bob's home page, what can she do? She can put her own certificate and signature block on the fake page, but when Alice reads the certificate she will immediately see that she is not talking to Bob because Bob's name is not in it. Trudy can modify Bob's home page on-the-fly, replacing Bob's public key with her own. However, when Alice runs the SHA-1 algorithm on the certificate, she will get a hash that does not agree with the one she gets when she applies the CA's well-known public key to the signature block. Since Trudy does not have the CA's private key, she has no way of generating a signature block that contains the hash of the modified Web page with her public key on it. In this way, Alice can be sure she has Bob's public key and not Trudy's or someone else's. And as we promised, this scheme does not require the CA to be on-line for verification, thus eliminating a potential bottleneck.

While the standard function of a certificate is to bind a public key to a principal, a certificate can also be used to bind a public key to an attribute. For example, a certificate could say: This public key belongs to someone over 18. It could be used to prove that the owner of the private key was not a minor and thus allowed to access material not suitable for children, and so on, but without disclosing the owner's identity. Typically, the person holding the certificate would send it to the Web site, principal, or process that cared about age. That site, principal, or process would then generate a random number and encrypt it with the public key in the certificate. If the owner were able to decrypt it and send it back, that would be proof that the owner indeed had the attribute stated in the certificate. Alternatively, the random number could be used to generate a session key for the ensuing conversation.

Another example of where a certificate might contain an attribute is in an object-oriented distributed system. Each object normally has multiple methods. The owner of the object could provide each customer with a certificate giving a bit map of which methods the customer is allowed to invoke and binding the bit map to a public key using a signed certificate. Again here, if the certificate holder can prove possession of the corresponding private key, he will be allowed to perform the methods in the bit map. It has the property that the owner's identity need not be known, a property useful in situations where privacy is important.

8.5.2 X.509

If everybody who wanted something signed went to the CA with a different kind of certificate, managing all the different formats would soon become a problem. To solve this problem, a standard for certificates has been devised and approved by ITU. The standard is called X.509 and is in widespread use on the Internet. It has gone through three versions since the initial standardization in 1988. We will discuss V3.

X.509 has been heavily influenced by the OSI world, borrowing some of its worst features (e.g., naming and encoding). Surprisingly, IETF went along with X.509, even though in nearly every other area, from machine addresses to transport protocols to e-mail formats, IETF generally ignored OSI and tried to do it right. The IETF version of X.509 is described in RFC 3280.

At its core, X.509 is a way to describe certificates. The primary fields in a certificate are listed in Figure 5.25. The descriptions given there should provide a general idea of what the fields do. For additional information, please consult the standard itself or RFC 2459.

Field	Meaning
Version	Which version of X.509
Serial number	This number plus the CA's name uniquely identifies the certificate
Signature algorithm	The algorithm used to sign the certificate
Issuer	X.500 name of the CA
Validity period	The starting and ending times of the validity period
Subject name	The entity whose key is being certified
Public key	The subject's public key and the ID of the algorithm using it
Issuer ID	An optional ID uniquely identifying the certificate's issuer
Subject ID	An optional ID uniquely identifying the certificate's subject
Extensions	Many extensions have been defined
Signature	The certificate's signature (signed by the CA's private key)

Figure 5.25. The basic fields of an X.509 certificate.

For example, if Bob works in the loan department of the Money Bank, his X.500 address might be:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

where C is for country, O is for organization, OU is for organizational unit, and CN is for common name. CAs and other entities are named in a similar

way. A substantial problem with X.500 names is that if Alice is trying to contact bob@moneybank.com and is given a certificate with an X.500 name, it may not be obvious to her that the certificate refers to the Bob she wants. Fortunately, starting with version 3, DNS names are now permitted instead of X.500 names, so this problem may eventually vanish.

Certificates are encoded using the OSI ASN.1 (Abstract Syntax Notation 1), which can be thought of as being like a struct in C, except with a very peculiar and verbose notation.

Public Key Infrastructures

Having a single CA to issue all the world's certificates obviously would not work. It would collapse under the load and be a central point of failure as well. A possible solution might be to have multiple CAs, all run by the same organization and all using the same private key to sign certificates. While this would solve the load and failure problems, it introduces a new problem: key leakage. If there were dozens of servers spread around the world, all holding the CA's private key, the chance of the private key being stolen or otherwise leaking out would be greatly increased. Since the compromise of this key would ruin the world's electronic security infrastructure, having a single central CA is very risky.

In addition, which organization would operate the CA? It is hard to imagine any authority that would be accepted worldwide as legitimate and trustworthy. In some countries people would insist that it be a government, while in other countries they would insist that it not be a government.

For these reasons, a different way for certifying public keys has evolved. It goes under the general name of PKI (Public Key Infrastructure). In this section we will summarize how it works in general, although there have been many proposals so the details will probably evolve in time.

A PKI has multiple components, including users, CAs, certificates, and directories. What the PKI does is provide a way of structuring these components and define standards for the various documents and protocols. A particularly simple form of PKI is a hierarchy of CAs, as depicted in Figure 5.26. In this example we have shown three levels, but in practice there might be fewer or more. The top-level CA, the root, certifies second-level CAs, which we call RAs (Regional Authorities) because they might cover some geographic region, such as a country or continent. This term is

not standard, though; in fact, no term is really standard for the different levels of the tree. These in turn certify the real CAs, which issue the X.509 certificates to organizations and individuals. When the root authorizes a new RA, it generates an X.509 certificate stating that it has approved the RA, includes the new RA's public key in it, signs it, and hands it to the RA. Similarly, when an RA approves a new CA, it produces and signs a certificate stating its approval and containing the CA's public key.

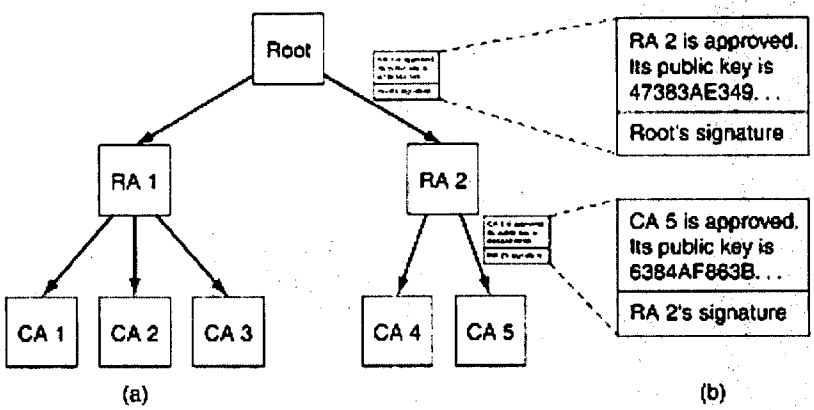


Figure 5.26. (a) A hierarchical PKI. (b) A chain of certificates.

Our PKI works like this. Suppose that Alice needs Bob's public key in order to communicate with him, so she looks for and finds a certificate containing it, signed by CA 5. But Alice has never heard of CA 5. For all she knows, CA 5 might be Bob's 10-year-old daughter. She could go to CA 5 and say: Prove your legitimacy. CA 5 responds with the certificate it got from RA 2, which contains CA 5's public key. Now armed with CA 5's public key, she can verify that Bob's certificate was indeed signed by CA 5 and is thus legal.

Unless RA 2 is Bob's 12-year-old son. So the next step is for her to ask RA 2 to prove it is legitimate. The response to her query is a certificate signed by the root and containing RA 2's public key. Now Alice is sure she has Bob's public key.

But how does Alice find the root's public key? Magic, it is assumed that everyone knows the root's public key. For example, her browser might have been shipped with the root's public key built in.

Bob is a friendly sort of guy and does not want to cause Alice a lot of work. He knows that she is going to have to check out CA 5 and RA 2, so to save

her some trouble, the collects the two needed certificates and gives her the two certificates along with his. Now she can use her own knowledge of the root's public key to verify the top-level certificate and the public key contained therein to verify the second one. In this way, Alice does not need to contact anyone to do the verification. Because the certificates are all signed, she can easily detect any attempts to tamper with their contents. A chain of certificates going back to the root like this is sometimes called a chain of trust or a certification path. The technique is widely used in practice.

Of course, we still have the problem of who is going to run the root. The solution is not to have a single root, but to have many roots, each with its own RAs and CAs. In fact, modern browsers come preloaded with the public keys for over 100 roots, sometimes referred to as trust anchors. In this way, having a single worldwide trusted authority can be avoided.

But there is now the issue of how the browser vendor decides which purported trust anchors are reliable and which are sleazy. It all comes down to the user trusting the browser vendor to make wise choices and not simply approve all trust anchors willing to pay its inclusion fee. Most browsers allow users to inspect the root keys (usually in the form of certificates signed by the root) and delete any that seem shady.

Directories

Another issue for any PKI is where certificates (and their chains back to some known trust anchor) are stored. One possibility is to have each user store his or her own certificates. While doing this is safe (i.e., there is no way for users to tamper with signed certificates without detection), it is also inconvenient. One alternative that has been proposed is to use DNS as a certificate directory. Before contacting Bob, Alice probably has to look up his IP address using DNS, so why not have DNS return Bob's entire certificate chain along with his IP address?

Some people think this is the way to go, but others would prefer dedicated directory servers whose only job is managing X.509 certificates. Such directories could provide lookup services by using properties of the X.500 names. For example, in theory such a directory service could answer a query such as: "Give me a list of all people named Alice who work in sales departments anywhere in the U.S. or Canada." LDAP might be a candidate for holding such information.

Revocation

The real world is full of certificates, too, such as passports and drivers' licenses. Sometimes these certificates can be revoked, for example, drivers' licenses can be revoked for drunken driving and other driving offenses. The same problem occurs in the digital world: the grantor of a certificate may decide to revoke it because the person or organization holding it has abused it in some way. It can also be revoked if the subject's private key has been exposed, or worse yet, the CA's private key has been compromised. Thus, a PKI needs to deal with the issue of revocation.

A first step in this direction is to have each CA periodically issue a CRL (Certificate Revocation List) giving the serial numbers of all certificates that it has revoked. Since certificates contain expiry times, the CRL need only contain the serial numbers of certificates that have not yet expired. Once its expiry time has passed, a certificate is automatically invalid, so no distinction is needed between those that just timed out and those that were actually revoked. In both cases, they cannot be used any more.

Unfortunately, introducing CRLs means that a user who is about to use a certificate must now acquire the CRL to see if the certificate has been revoked. If it has been, it should not be used. However, even if the certificate is not on the list, it might have been revoked just after the list was published. Thus, the only way to really be sure is to ask the CA. And on the next use of the same certificate, the CA has to be asked again, since the certificate might have been revoked a few seconds ago.

Another complication is that a revoked certificate could conceivably be reinstated, for example, if it was revoked for nonpayment of some fee that has since been paid. Having to deal with revocation (and possibly reinstatement) eliminates one of the best properties of certificates, namely, that they can be used without having to contact a CA.

Where should CRLs be stored? A good place would be the same place the certificates themselves are stored. One strategy is for the CA to actively push out CRLs periodically and have the directories process them by simply removing the revoked certificates. If directories are not used for storing certificates, the CRLs can be cached at various convenient places around the network. Since a CRL is itself a signed document, if it is tampered with, that tampering can be easily detected.

If certificates have long lifetimes, the CRLs will be long, too. For example, if credit cards are valid for 5 years, the number of revocations outstanding will be much longer than if new cards are issued every 3 months. A standard way to deal with long CRLs is to issue a master list infrequently, but issue updates to it more often. Doing this reduces the bandwidth needed for distributing the CRLs.

5.6 Communication Security

We have now finished our study of the tools of the trade. Most of the important techniques and protocols have been covered. The rest of the chapter is about how these techniques are applied in practice to provide network security, plus some thoughts about the social aspects of security at the end of the chapter.

In the following four sections, we will look at communication security, that is, how to get the bits secretly and without modification from source to destination and how to keep unwanted bits outside the door. These are by no means the only security issues in networking, but they are certainly among the most important ones, making this a good place to start.

IPsec

IETF has known for years that security was lacking in the Internet. Adding it was not easy because a war broke out about where to put it. Most security experts believe that to be really secure, encryption and integrity checks have to be end to end (i.e., in the application layer). That is, the source process encrypts and/or integrity protects the data and sends that to the destination process where it is decrypted and/or verified. Any tampering done in between these two processes, including within either operating system, can then be detected. The trouble with this approach is that it requires changing all the applications to make them security aware. In this view, the next best approach is putting encryption in the transport layer or in a new layer between the application layer and the transport layer, making it still end to end but not requiring applications to be changed.

The opposite view is that users do not understand security and will not be capable of using it correctly and nobody wants to modify existing programs in any way, so the network layer should authenticate and/or encrypt packets without the users being involved. After years of pitched battles, this view won enough support that a network layer security standard was

defined. In part the argument was that having network layer encryption does not prevent security-aware users from doing it right and it does help security-unaware users to some extent.

The result of this war was a design called IPsec (IP security), which is described in RFCs 2401, 2402, and 2406, among others. Not all users want encryption (because it is computationally expensive). Rather than make it optional, it was decided to require encryption all the time but permit the use of a null algorithm. The null algorithm is described and praised for its simplicity, ease of implementation, and great speed in RFC 2410.

The complete IPsec design is a framework for multiple services, algorithms and granularities. The reason for multiple services is that not everyone wants to pay the price for having all the services all the time, so the services are available a la carte. The major services are secrecy, data integrity, and protection from replay attacks (intruder replays a conversation). All of these are based on symmetric-key cryptography because high performance is crucial.

The reason for having multiple algorithms is that an algorithm that is now thought to be secure may be broken in the future. By making IPsec algorithm-independent, the framework can survive even if some particular algorithm is later broken.

The reason for having multiple granularities is to make it possible to protect a single TCP connection, all traffic between a pair of hosts, or all traffic between a pair of secure routers, among other possibilities.

One slightly surprising aspect of IPsec is that even though it is in the IP layer, it is connection oriented. Actually, that is not so surprising because to have any security, a key must be established and used for some period of time in essence, a kind of connection. Also connections amortize the setup costs over many packets. A "connection" in the context of IPsec is called an SA (security association). An SA is a simplex connection between two end points and has a security identifier associated with it. If secure traffic is needed in both directions, two security associations are required. Security identifiers are carried in packets traveling on these secure connections and are used to look up keys and other relevant information when a secure packet arrives.

Technically, IPsec has two principal parts. The first part describes two new headers that can be added to packets to carry the security identifier, integrity control data, and other information. The other part, ISAKMP (Internet Security Association and Key Management Protocol) deals with establishing keys. We will not deal with ISAKMP further because (1) it is extremely complex and (2) its main protocol, IKE (Internet Key Exchange), is deeply flawed and needs to be replaced.

IPsec can be used in either of two modes. In transport mode, the IPsec header is inserted just after the IP header. The Protocol field in the IP header is changed to indicate that an IPsec header follows the normal IP header (before the TCP header). The IPsec header contains security information, primarily the SA identifier, a new sequence number, and possibly an integrity check of the payload.

In tunnel mode, the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header. Tunnel mode is useful when the tunnel ends at a location other than the final destination. In some cases, the end of the tunnel is a security gateway machine, for example, a company firewall. In this mode, the firewall encapsulates and decapsulates packets as they pass though the firewall. By terminating the tunnel at this secure machine, the machines on the company LAN do not have to be aware of IPsec. Only the firewall has to know about it.

Tunnel mode is also useful when a bundle of TCP connections is aggregated and handled as one encrypted stream because it prevents an intruder from seeing who is sending how many packets to whom. Sometimes just knowing how much traffic is going where is valuable information. For example, if during a military crisis, the amount of traffic flowing between the Pentagon and the White House drops sharply, but the amount of traffic between the Pentagon and some military installation deep in the Colorado Rocky Mountains increases by the same amount, an intruder might be able to deduce some useful information from this data. Studying the flow patterns of packets, even if they are encrypted, is called traffic analysis. Tunnel mode provides a way to foil it to some extent. The disadvantage of tunnel mode is that it adds an extra IP header, thus increasing packet size substantially. In contrast, transport mode does not affect packet size as much.

The first new header is AH (Authentication Header). It provides integrity checking and antireplay security, but not secrecy (i.e., no data encryption).

The use of AH in transport mode is illustrated in Figure 5.27. In IPv4 it is interposed between the IP header (including any options) and the TCP header. In IPv6 it is just another extension header and treated as such. In fact, the format is close to that of a standard IPv6 extension header. The payload may have to be padded out to some particular length for the authentication algorithm, as shown.

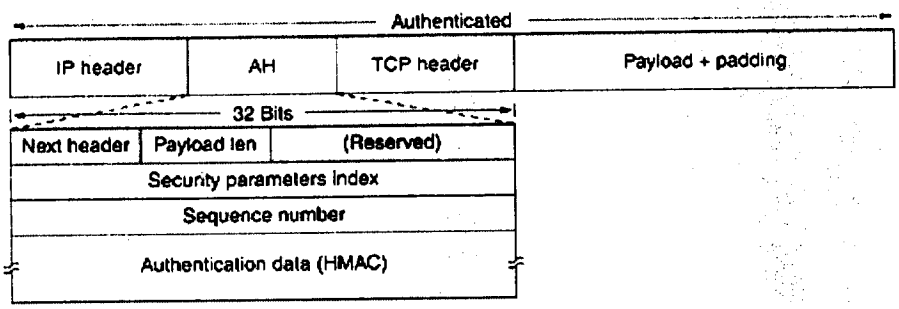


Figure 5.27. The IPsec authentication header in transport mode for IPv4.

Let us now examine the AH header. The Next header field is used to store the previous value that the IP Protocol field had before it was replaced with 51 to indicate that an AH header follows. In most cases, the code for TCP (6) will go here. The Payload length is the number of 32-bit words in the AH header minus 2.

The Security parameters index is the connection identifier. It is inserted by the sender to indicate a particular record in the receiver's database. This record contains the shared key used on this connection and other information about the connection. If this protocol had been invented by ITU rather than IETF, this field would have been called Virtual circuit number.

The Sequence number field is used to number all the packets sent on an SA. Every packet gets a unique number, even retransmissions. In other words, the retransmission of a packet gets a different number here than the original (even though its TCP sequence number is the same). The purpose of this field is to detect replay attacks. These sequence numbers may not wrap around. If all 2^{32} are exhausted, a new SA must be established to continue communication.

Finally, we come to the Authentication data, which is a variable-length field that contains the payload's digital signature. When the SA is established, the two sides negotiate which signature algorithm they are going to use. Normally, public-key cryptography is not used here because packets must

be processed extremely rapidly and all known public-key algorithms are too slow. Since IPsec is based on symmetric-key cryptography and the sender and receiver negotiate a shared key before setting up an SA, the shared key is used in the signature computation. One simple way is to compute the hash over the packet plus the shared key. The shared key is not transmitted, of course. A scheme like this is called an HMAC (Hashed Message Authentication Code). It is much faster to compute than first running SHA-1 and then running RSA on the result.

The AH header does not allow encryption of the data, so it is mostly useful when integrity checking is needed but secrecy is not needed. One noteworthy feature of AH is that the integrity check covers some of the fields in the IP header, namely, those that do not change as the packet moves from router to router. The Time to live field changes on each hop, for example, so it cannot be included in the integrity check. However, the IP source address is included in the check, making it impossible for an intruder to falsify the origin of a packet.

The alternative IPsec header is ESP (Encapsulating Security Payload). Its use for both transport mode and tunnel mode is shown in Figure 5.28.

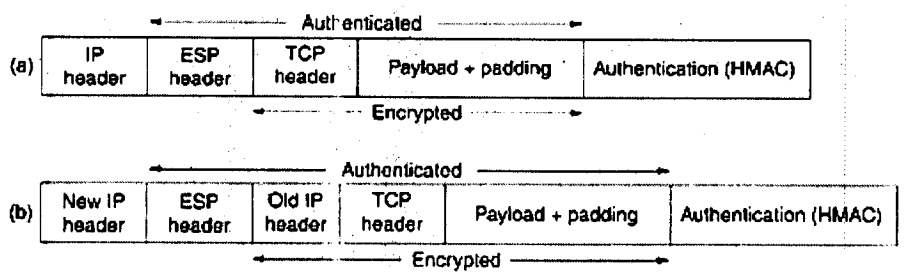


Figure 5.28. (a) ESP in transport mode. (b) ESP in tunnel mode.

The ESP header consists of two 32-bit words. They are the Security parameters index and Sequence number fields that we saw in AH. A third word that generally follows them (but is technically not part of the header) is the Initialization vector used for the data encryption, unless null encryption is used, in which case it is omitted.

ESP also provides for HMAC integrity checks, as does AH, but rather than being included in the header, they come after the payload, as shown in Figure 5.28. Putting the HMAC at the end has an advantage in a hardware implementation. The HMAC can be calculated as the bits are going out

over the network interface and appended to the end. This is why Ethernet and other LANs have their CRCs in a trailer, rather than in a header. With AH, the packet has to be buffered and the signature computed before the packet can be sent, potentially reducing the number of packets/sec that can be sent.

Given that ESP can do everything AH can do and more and is more efficient to boot, the question arises: Why bother having AH at all? The answer is mostly historical. Originally, AH handled only integrity and ESP handled only secrecy. Later, integrity was added to ESP, but the people who designed AH did not want to let it die after all that work. Their only real argument, however, is that AH checks part of the IP header, which ESP does not, but it is a weak argument. Another weak argument is that a product supporting AH but not ESP might have less trouble getting an export license because it cannot do encryption. AH is likely to be phased out in the future.

Firewalls

The ability to connect any computer, anywhere, to any other computer, anywhere, is a mixed blessing. For individuals at home, wandering around the Internet is lots of fun. For corporate security managers, it is a nightmare. Most companies have large amounts of confidential information on-line trade secrets, product development plans, marketing strategies, financial analyses, etc. Disclosure of this information to a competitor could have dire consequences.

In addition to the danger of information leaking out, there is also a danger of information leaking in. In particular, viruses, worms, and other digital pests can breach security, destroy valuable data, and waste large amounts of administrators' time trying to clean up the mess they leave. Often they are imported by careless employees who want to play some nifty new game.

Consequently, mechanisms are needed to keep "good" bits in and "bad" bits out. One method is to use IPsec. This approach protects data in transit between secure sites. However, IPsec does nothing to keep digital pests and intruders from getting onto the company LAN. To see how to accomplish this goal, we need to look at firewalls.

Firewalls are just a modern adaptation of that old medieval security standby: digging a deep moat around your castle. This design forced everyone entering or leaving the castle to pass over a single drawbridge, where they could be inspected by the I/O police. With networks, the same trick is possible: a company can have many LANs connected in arbitrary ways, but all traffic to or from the company is forced through an electronic drawbridge (firewall), as shown in Figure 5.29.

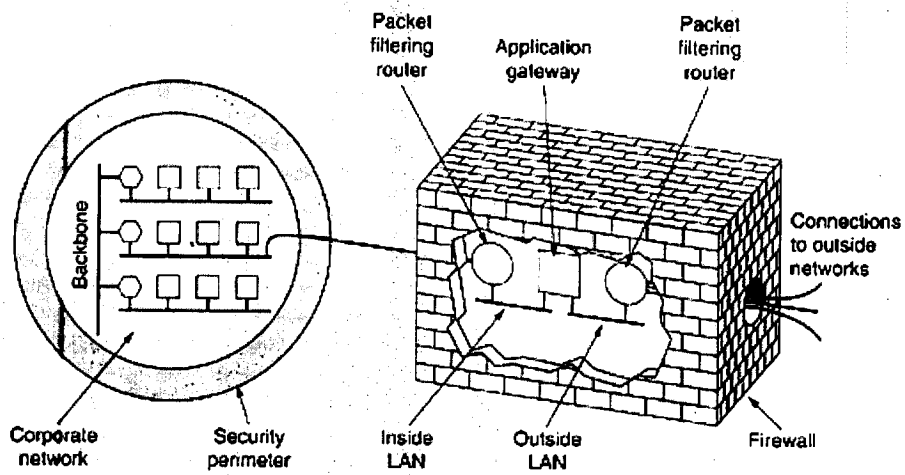


Figure 5.29. A firewall consisting of two packet filters and an application gateway.

The firewall in this configuration has two components: two routers that do packet filtering and an application gateway. Simpler configurations also exist, but the advantage of this design is that every packet must transit two filters and an application gateway to go in or out. No other route exists. Readers who think that one security checkpoint is enough clearly have not made an international flight on a scheduled airline recently.

Each packet filter is a standard router equipped with some extra functionality. The extra functionality allows every incoming or outgoing packet to be inspected. Packets meeting some criterion are forwarded normally. Those that fail the test are dropped.

In Figure 5.29, most likely the packet filter on the inside LAN checks outgoing packets and the one on the outside LAN checks incoming packets. Packets crossing the first hurdle go to the application gateway for further examination. The point of putting the two packet filters on different

LANs is to ensure that no packet gets in or out without having to pass through the application gateway: there is no path around it.

Packet filters are typically driven by tables configured by the system administrator. These tables list sources and destinations that are acceptable, sources and destinations that are blocked, and default rules about what to do with packets coming from or going to other machines.

In the common case of a TCP/IP setting, a source or destination consists of an IP address and a port. Ports indicate which service is desired. For example, TCP port 23 is for telnet, TCP port 79 is for finger, and TCP port 119 is for USENET news. A company could block incoming packets for all IP addresses combined with one of these ports. In this way, no one outside the company could log in via telnet or look up people by using the Finger daemon. Furthermore, the company would be spared from having employees spend all day reading USENET news.

Blocking outgoing packets is trickier because although most sites stick to the standard port numbering conventions, they are not forced to do so. Furthermore, for some important services, such as FTP (File Transfer Protocol), port numbers are assigned dynamically. In addition, although blocking TCP connections is difficult, blocking UDP packets is even harder because so little is known a priori about what they will do. Many packet filters are configured to simply ban UDP traffic altogether.

The second half of the firewall is the application gateway. Rather than just looking at raw packets, the gateway operates at the application level. A mail gateway, for example, can be set up to examine each message going in or coming out. For each one, the gateway decides whether to transmit or discard the message based on header fields, message size, or even the content (e.g., at a military installation, the presence of words like "nuclear" or "bomb" might cause some special action to be taken).

Installations are free to set up one or more application gateways for specific applications, but it is not uncommon for suspicious organizations to permit e-mail in and out, and perhaps permit use of the World Wide Web, but to ban everything else as too dicey. Combined with encryption and packet filtering, this arrangement offers a limited amount of security at the cost of some inconvenience.

Even if the firewall is perfectly configured, plenty of security problems still exist. For example, if a firewall is configured to allow in packets from only specific networks (e.g., the company's other plants), an intruder outside the firewall can put in false source addresses to bypass this check. If an insider wants to ship out secret documents, he can encrypt them or even photograph them and ship the photos as JPEG files, which by passes any word filters. And we have not even discussed the fact that 70% of all attacks come from inside the firewall, for example, from disgruntled employees.

In addition, there is a whole other class of attacks that firewalls cannot deal with. The basic idea of a firewall is to prevent intruders from getting in and secret data from getting out. Unfortunately, there are people who have nothing better to do than try to bring certain sites down. They do this by sending legitimate packets at the target in great numbers until it collapses under the load. For example, to cripple a Web site, an intruder can send a TCP SYN packet to establish a connection. The site will then allocate a table slot for the connection and send a SYN + ACK packet in reply. If the intruder does not respond, the table slot will be tied up for a few seconds until it times out. If the intruder sends thousands of connection requests, all the table slots will fill up and no legitimate connections will be able to get through. Attacks in which the intruder's goal is to shut down the target rather than steal data are called DoS (Denial of Service) attacks. Usually, the request packets have false source addresses so the intruder cannot be traced easily.

An even worse variant is one in which the intruder has already broken into hundreds of computers elsewhere in the world, and then commands all of them to attack the same target at the same time. Not only does this approach increase the intruder's firepower, it also reduces his chance of detection, since the packets are coming from a large number of machines belonging to unsuspecting users. Such an attack is called a DDoS (Distributed Denial of Service) attack. This attack is difficult to defend against. Even if the attacked machine can quickly recognize a bogus request, it does take some time to process and discard the request, and if enough requests per second arrive, the CPU will spend all its time dealing with them.

Virtual Private Networks

Many companies have offices and plants scattered over many cities, sometimes over multiple countries. In the olden days, before public data networks, it was common for such companies to lease lines from the telephone company between some or all pairs of locations. Some companies still do this. A network built up from company computers and leased telephone lines is called a private network. An example private network connecting three locations is shown in Figure 5.30(a)

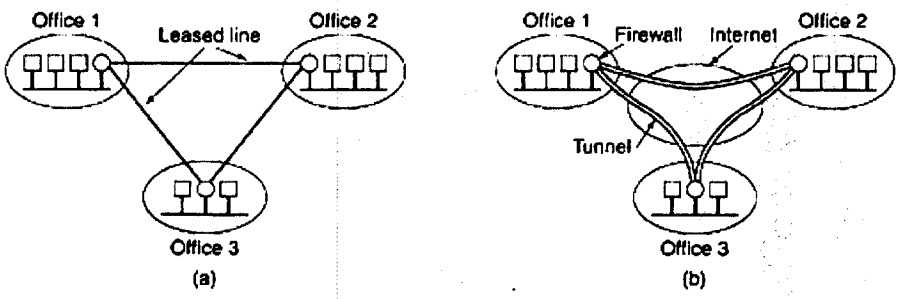


Figure 5.30. (a) A leased-line private network. (b) A virtual private network.

Private networks work fine and are very secure. If the only lines available are the leased lines, no traffic can leak out of company locations and intruders have to physically wiretap the lines to break in, which is not easy to do. The problem with private networks is that leasing a single T1 line costs thousands of dollars a month and T3 lines are many times more expensive. When public data networks and later the Internet appeared, many companies wanted to move their data (and possibly voice) traffic to the public network, but without giving up the security of the private network.

This demand soon led to the invention of VPNs (Virtual Private Networks), which are overlay networks on top of public networks but with most of the properties of private networks. They are called "virtual" because they are merely an illusion, just as virtual circuits are not real circuits and virtual memory is not real memory.

Although VPNs can be implemented on top of ATM (or frame relay), an increasingly popular approach is to build VPNs directly over the Internet. A common design is to equip each office with a firewall and create tunnels through the Internet between all pairs of offices, as illustrated in Figure 5.30(b). If IPsec is used for the tunneling, then it is possible to aggregate all

traffic between any two pairs of offices onto a single authenticated, encrypted SA, thus providing integrity control, secrecy, and even considerable immunity to traffic analysis.

When the system is brought up, each pair of firewalls has to negotiate the parameters of its SA, including the services, modes, algorithms, and keys. Many firewalls have VPN capabilities built in, although some ordinary routers can do this as well. But since firewalls are primarily in the security business, it is natural to have the tunnels begin and end at the firewalls, providing a clear separation between the company and the Internet. Thus, firewalls, VPNs, and IPsec with ESP in tunnel mode are a natural combination and widely used in practice.

Once the SAs have been established, traffic can begin flowing. To a router within the Internet, a packet traveling along a VPN tunnel is just an ordinary packet. The only thing unusual about it is the presence of the IPsec header after the IP header, but since these extra headers have no effect on the forwarding process, the routers do not care about this extra header.

A key advantage of organizing a VPN this way is that it is completely transparent to all user software. The firewalls set up and manage the SAs. The only person who is even aware of this setup is the system administrator who has to configure and manage the firewalls. To everyone else, it is like having a leased-line private network again. For more about VPNs, see (Brown, 1999; and Izzo, 2000).

Wireless Security

It is surprisingly easy to design a system that is logically completely secure by using VPNs and firewalls, but that, in practice, leaks like a sieve. This situation can occur if some of the machines are wireless and use radio communication, which passes right over the firewall in both directions. The range of 802.11 networks is often a few hundred meters, so anyone who wants to spy on a company can simply drive into the employee parking lot in the morning, leave an 802.11-enabled notebook computer in the car to record everything it hears, and take off for the day. By late afternoon, the hard disk will be full of valuable goodies. Theoretically, this leakage is not supposed to happen. Theoretically, people are not supposed to rob banks, either.

Much of the security problem can be traced to the manufacturers of wireless base stations (access points) trying to make their products user friendly. Usually, if the user takes the device out of the box and plugs it into the electrical power socket, it begins operating immediately—nearly always with no security at all, blurting secrets to everyone within radio range. If it is then plugged into an Ethernet, all the Ethernet traffic suddenly appears in the parking lot as well. Wireless is a snooper's dream come true: free data without having to do any work. It therefore goes without saying that security is even more important for wireless systems than for wired ones. In this section, we will look at some ways wireless networks handle security. Some additional information can be found in.

802.11 Security

The 802.11 standard prescribes a data link-level security protocol called WEP (Wired Equivalent Privacy), which is designed to make the security of a wireless LAN as good as that of a wired LAN. Since the default for wired LANs is no security at all, this goal is easy to achieve, and WEP achieves it, as we shall see.

When 802.11 security is enabled, each station has a secret key shared with the base station. How the keys are distributed is not specified by the standard. They could be preloaded by the manufacturer. They could be exchanged in advance over the wired network. Finally, either the base station or user machine could pick a random key and send it to the other one over the air encrypted with the other one's public key. Once established, keys generally remain stable for months or years.

WEP encryption uses a stream cipher based on the RC4 algorithm. RC4 was designed by Ronald Rivest and kept secret until it leaked out and was posted to the Internet in 1994. As we have pointed out before, it is nearly impossible to keep algorithms secret, even when the goal is guarding intellectual property (as it was in this case) rather than security by obscurity (which was not the goal with RC4). In WEP, RC4 generates a keystream that is XORed with the plaintext to form the ciphertext.

Each packet payload is encrypted using the method of Figure 5.31. First the payload is checksummed using the CRC-32 polynomial and the checksum appended to the payload to form the plaintext for the encryption algorithm. Then this plaintext is XORed with a chunk of keystream its own size. The result is the ciphertext. The IV used to start RC4 is sent along

with the ciphertext. When the receiver gets the packet, it extracts the encrypted payload from it, generates the keystream from the shared secret key and the IV it just got, and XORs the keystream with the payload to recover the plaintext. It can then verify the checksum to see if the packet has been tampered with.

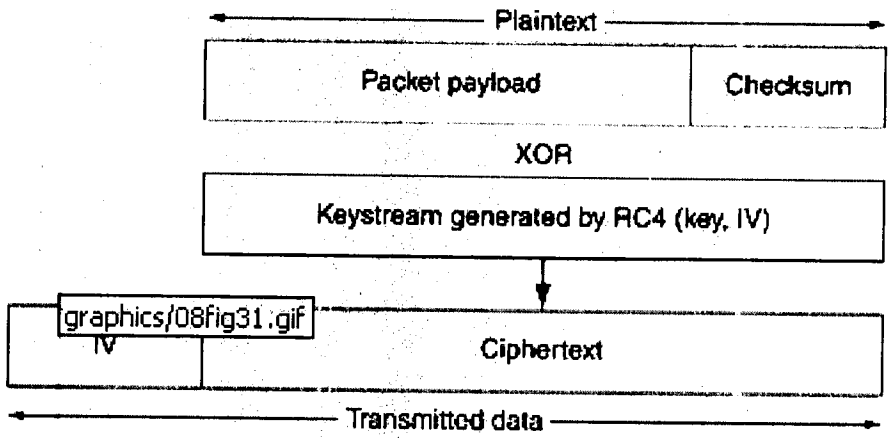


Figure 5.31. Packet encryption using WEP.

While this approach looks good at first glance, a method for breaking it has already been published (Borisov et al., 2001). Below we will summarize their results. First of all, surprisingly many installations use the same shared key for all users, in which case each user can read all the other users' traffic. This is certainly equivalent to Ethernet, but it is not very secure.

But even if each user has a distinct key, WEP can still be attacked. Since keys are generally stable for long periods of time, the WEP standard recommends (but does not mandate) that IV be changed on every packet to avoid the keystream reuse attack. Unfortunately, many 802.11 cards for notebook computers reset IV to 0 when the card is inserted into the computer, and increment it by one on each packet sent. Since people often remove and reinsert these cards, packets with low IV values are common. If Trudy can collect several packets sent by the same user with the same IV value (which is itself sent in plaintext along with each packet), she can compute the XOR of two plaintext values and probably break the cipher that way.

But even if the 802.11 card picks a random IV for each packet, the IVs are only 24 bits, so after 2^{24} packets have been sent, they have to be reused.

Worse yet, with randomly chosen IVs, the expected number of packets that have to be sent before the same one is used twice is about 5000, due to the birthday attack. Thus, if Trudy listens for a few minutes, she is almost sure to capture two packets with the same IV and same key. By XORing the ciphertexts she is able to obtain the XOR of the plaintexts. This bit sequence can be attacked in various ways to recover the plaintexts. With some more work, the keystream for that IV can also be obtained. Trudy can continue working like this for a while and compile a dictionary of keystreams for various IVs. Once an IV has been broken, all the packets sent with it in the future (but also in the past) can be fully decrypted.

Furthermore, since IVs are used at random, once Trudy has determined a valid (IV, keystream) pair, she can use it to generate all the packets she wants to using it and thus actively interfere with communication. Theoretically, a receiver could notice that large numbers of packets suddenly all have the same IV, but (1) WEP allows this, and (2) nobody checks for this anyway.

Finally, the CRC is not worth much, since it is possible for Trudy to change the payload and make the corresponding change to the CRC, without even having to remove the encryption In short, breaking 802.11's security is fairly straightforward, and we have not even listed all the attacks Borisov et al. found.

In August 2001, a month after the Borisov et al. paper was presented, another devastating attack on WEP was published. This one found cryptographic weaknesses in RC4 itself. Fluhrer et al. discovered that many of the keys have the property that it is possible to derive some key bits from the keystream. If this attack is performed repeatedly, it is possible to derive the entire key with a modest amount of effort. Being somewhat theoretically inclined, Fluhrer et al. did not actually try to break any 802.11 LANs.

In contrast, when a summer student and two researchers at AT&T Labs learned about the Fluhrer et al. attack, they decided to try it out for real. Within a week they had broken their first 128-bit key on a production 802.11 LAN, and most of the week was actually devoted to looking for the cheapest 802.11 card they could find, getting permission to buy it, installing it, and testing it. The programming took only two hours.

When they announced their results, CNN ran a story entitled "Off-the-Shelf Hack Breaks Wireless Encryption," in which some industry gurus tried to

pooh-pooh their results by saying what they had done was trivial, given the Fluhrer et al. results. While that remark is technically true, the fact remains that the combined efforts of these two teams demonstrated a fatal flaw in WEP and 802.11.

On September 7, 2001, IEEE responded to the fact that WEP was now completely broken by issuing a short statement making six points that can be roughly summarized as follows:

1. We told you that WEP security was no better than Ethernet's.
2. A much bigger threat is forgetting to enable security at all.
3. Try using some other security (e.g., transport layer security).
4. The next version, 802.11i, will have better security.
5. Future certification will mandate the use of 802.11i.
6. We will try to figure out what to do until 802.11i arrives.

We have gone through this story in some detail to make the point that getting security right is not easy, even for experts.

Bluetooth Security

Bluetooth has a considerably shorter range than 802.11, so it cannot be attacked from the parking lot, but security is still an issue here. For example, imagine that Alice's computer is equipped with a wireless Bluetooth keyboard. In the absence of security, if Trudy happened to be in the adjacent office, she could read everything Alice typed in, including all her outgoing e-mail. She could also capture everything Alice's computer sent to the Bluetooth printer sitting next to it (e.g., incoming e-mail and confidential reports). Fortunately, Bluetooth has an elaborate security scheme to try to foil the world's Trudies. We will now summarize the main features of it below.

Bluetooth has three security modes, ranging from nothing at all to full data encryption and integrity control. As with 802.11, if security is disabled (the default), there is no security. Most users have security turned off until a serious breach has occurred; then they turn it on. In the agricultural world, this approach is known as locking the barn door after the horse has escaped.

Bluetooth provides security in multiple layers. In the physical layer, frequency hopping provides a tiny bit of security, but since any Bluetooth device that moves into a piconet has to be told the frequency hopping sequence, this sequence is obviously not a secret. The real security starts when the newly-arrived slave asks for a channel with the master. The two devices are assumed to share a secret key set up in advance. In some cases, both are hardwired by the manufacturer (e.g., for a headset and mobile phone sold as a unit). In other cases, one device (e.g., the headset) has a hardwired key and the user has to enter that key into the other device (e.g., the mobile phone) as a decimal number. These shared keys are called passkeys.

To establish a channel, the slave and master each check to see if the other one knows the passkey. If so, they negotiate whether that channel will be encrypted, integrity controlled, or both. Then they select a random 128-bit session key, some of whose bits may be public. The point of allowing this key weakening is to comply with government restrictions in various countries designed to prevent the export or use of keys longer than the government can break.

Encryption uses a stream cipher called E_0; integrity control uses SAFER+. Both are traditional symmetric-key block ciphers. SAFER+ was submitted to the AES bake-off, but was eliminated in the first round because it was slower than the other candidates. Bluetooth was finalized before the AES cipher was chosen; otherwise it would most likely have used Rijndael.

The actual encryption using the stream cipher is shown in Figure 5.14, with the plaintext XORed with the keystream to generate the ciphertext. Unfortunately, E_0 itself (like RC4) may have fatal weaknesses. While it was not broken at the time of this writing, its similarities to the A5/1 cipher, whose spectacular failure compromises all GSM telephone traffic, are cause for concern (Biryukov et al., 2000). It sometimes amazes people (including the author), that in the perennial cat-and-mouse game between cryptographers and cryptanalysts, the cryptanalysts are so often on the winning side.

Another security issue is that Bluetooth authenticates only devices, not users, so theft of a Bluetooth device may give the thief access to the user's financial and other accounts. However, Bluetooth also implements security in the upper layers, so even in the event of a breach of link-level security, some security may remain, especially for applications that require a PIN

code to be entered manually from some kind of keyboard to complete the transaction.

WAP 2.0 Security

For the most part, the WAP Forum learned its lesson from having a nonstandard protocol stack in WAP 1.0. WAP 2.0 largely uses standard protocols in all layers. Security is no exception. Since it is IP based, it supports full use of IPsec in the network layer. In the transport layer, TCP connections can be protected by TLS, an IETF standard we will study later in this chapter. Higher still, it uses HTTP client authentication, as defined in RFC 2617. Application-layer crypto libraries provide for integrity control and nonrepudiation. All in all, since WAP 2.0 is based on well-known standards, there is a chance that its security services, in particular, privacy, authentication, integrity control, and nonrepudiation may fare better than 802.11 and Bluetooth security.

5.7 Authentication Protocols

Authentication is the technique by which a process verifies that its communication partner is who it is supposed to be and not an imposter. Verifying the identity of a remote process in the face of a malicious, active intruder is surprisingly difficult and requires complex protocols based on cryptography. In this section, we will study some of the many authentication protocols that are used on insecure computer networks.

As an aside, some people confuse authorization with authentication. Authentication deals with the question of whether you are actually communicating with a specific process. Authorization is concerned with what that process is permitted to do. For example, a client process contacts a file server and says: I am Scott's process and I want to delete the file cookbook.old. From the file server's point of view, two questions must be answered:

1. Is this actually Scott's process (authentication)?
2. Is Scott allowed to delete cookbook.old (authorization)?

Only after both of these questions have been unambiguously answered in the affirmative can the requested action take place. The former question is really the key one. Once the file server knows to whom it is talking, checking authorization is just a matter of looking up entries in local tables

or databases. For this reason, we will concentrate on authentication in this section.

The general model that all authentication protocols use is this. Alice starts out by sending a message either to Bob or to a trusted KDC (Key Distribution Center), which is expected to be honest. Several other message exchanges follow in various directions. As these messages are being sent Trudy may intercept, modify, or replay them in order to trick Alice and Bob or just to gum up the works.

Nevertheless, when the protocol has been completed, Alice is sure she is talking to Bob and Bob is sure he is talking to Alice. Furthermore, in most of the protocols, the two of them will also have established a secret session key for use in the upcoming conversation. In practice, for performance reasons, all data traffic is encrypted using symmetric-key cryptography (typically AES or triple DES), although public-key cryptography is widely used for the authentication protocols themselves and for establishing the session key.

The point of using a new, randomly-chosen session key for each new connection is to minimize the amount of traffic that gets sent with the users' secret keys or public keys, to reduce the amount of ciphertext an intruder can obtain, and to minimize the damage done if a process crashes and its core dump falls into the wrong hands. Hopefully, the only key present then will be the session key. All the permanent keys should have been carefully zeroed out after the session was established.

Authentication Based on a Shared Secret Key

For our first authentication protocol, we will assume that Alice and Bob already share a secret key, K_{AB} . This shared key might have been agreed upon on the telephone or in person, but, in any event, not on the (insecure) network.

This protocol is based on a principle found in many authentication protocols: one party sends a random number to the other, who then transforms it in a special way and then returns the result. Such protocols are called challenge-response protocols. In this and subsequent authentication protocols, the following notation will be used:

A, B are the identities of Alice and Bob.

R_i's are the challenges, where the subscript identifies the challenger.

K_i are keys, where i indicates the owner.

K_S is the session key.

The message sequence for our first shared-key authentication protocol is illustrated in Figure 5.32. In message 1, Alice sends her identity, A, to Bob in a way that Bob understands. Bob, of course, has no way of knowing whether this message came from Alice or from Trudy, so he chooses a challenge, a large random number, R_B, and sends it back to "Alice" as message 2, in plaintext. Random numbers used just once in challenge-response protocols like this one are called nonces. Alice then encrypts the message with the key she shares with Bob and sends the ciphertext, K_{AB} (R_B), back in message 3. When Bob sees this message, he immediately knows that it came from Alice because Trudy does not know K_{AB} and thus could not have generated it. Furthermore, since R_B was chosen randomly from a large space (say, 128-bit random numbers), it is very unlikely that Trudy would have seen R_B and its response from an earlier session. It is equally unlikely that she could guess the correct response to any challenge.

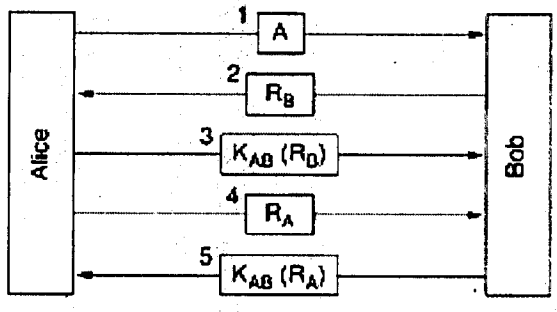


Figure 5.32. Two-way authentication using a challenge-response protocol.

At this point, Bob is sure he is talking to Alice, but Alice is not sure of anything. For all Alice knows, Trudy might have intercepted message 1 and sent back R_B in response. Maybe Bob died last night. To find out to whom she is talking, Alice picks a random number, R_A and sends it to Bob as plaintext, in message 4. When Bob responds with K_{AB} (R_A), Alice knows she is talking to Bob. If they wish to establish a session key now, Alice can pick one, K_S, and send it to Bob encrypted with K_{AB}.

The protocol of Figure 5.32 contains five messages. Let us see if we can be clever and eliminate some of them. One approach is illustrated in Figure 5.33. Here Alice initiates the challenge-response protocol instead of waiting for Bob to do it. Similarly, while he is responding to Alice's challenge, Bob sends his own. The entire protocol can be reduced to three messages instead of five.

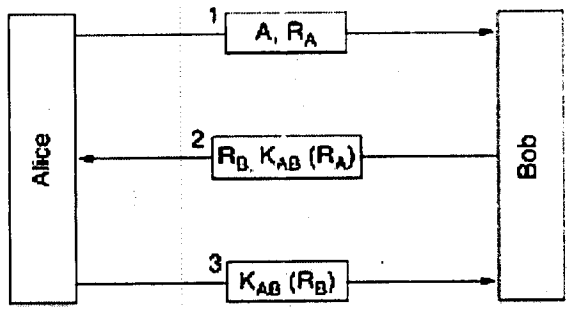


Figure 5.33. A shortened two-way authentication protocol.

Is this new protocol an improvement over the original one? In one sense it is: it is shorter. Unfortunately, it is also wrong. Under certain circumstances, Trudy can defeat this protocol by using what is known as a reflection attack. In particular, Trudy can break it if it is possible to open multiple sessions with Bob at once. This situation would be true, for example, if Bob is a bank and is prepared to accept many simultaneous connections from teller machines at once.

Trudy's reflection attack is shown in Figure 5.34. It starts out with Trudy claiming she is Alice and sending R_T. Bob responds, as usual, with his own challenge, R_B. Now Trudy is stuck. What can she do? She does not know K_{AB} (R_B).

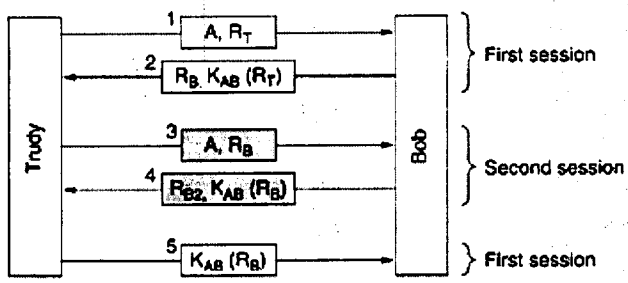


Figure 5.34. The reflection attack.

She can open a second session with message 3, supplying the R_B taken from message 2 as her challenge. Bob calmly encrypts it and sends back

K_{AB} (R_B)in message 4. We have shaded the messages on the second session to make them stand out. Now Trudy has the missing information, so she can complete the first session and abort the second one. Bob is now convinced that Trudy is Alice, so when she asks for her bank account balance, he gives it to her without question. Then when she asks him to transfer it all to a secret bank account in Switzerland, he does so without a moment's hesitation.

The moral of this story is:

Designing a correct authentication protocol is harder than it looks.

The following four general rules often help:

1. Have the initiator prove who she is before the responder has to. In this case, Bob gives away valuable information before Trudy has to give any evidence of who she is.

2. Have the initiator and responder use different keys for proof, even if this means having two shared keys, K_{AB} and K'_{AB} .

3. Have the initiator and responder draw their challenges from different sets. For example, the initiator must use even numbers and the responder must use odd numbers.

4. Make the protocol resistant to attacks involving a second parallel session in which information obtained in one session is used in a different one.

If even one of these rules is violated, the protocol can frequently be broken. Here, all four rules were violated, with disastrous consequences.

Now let us go back and take a closer look at Figure 5.32 Surely that protocol is not subject to a reflection attack? Well, it depends. It is quite subtle. Trudy was able to defeat our protocol by using a reflection attack because it was possible to open a second session with Bob and trick him into answering his own questions. What would happen if Alice were a general-purpose computer that also accepted multiple sessions, rather than a person at a computer? Let us take a look what Trudy can do.

To see how Trudy's attack works, see Figure 5.35. Alice starts out by announcing her identity in message 1. Trudy intercepts this message and

begins her own session with message 2, claiming to be Bob. Again we have shaded the session 2 messages. Alice responds to message 2 by saying: You claim to be Bob? Prove it. in message 3. At this point Trudy is stuck because she cannot prove she is Bob.

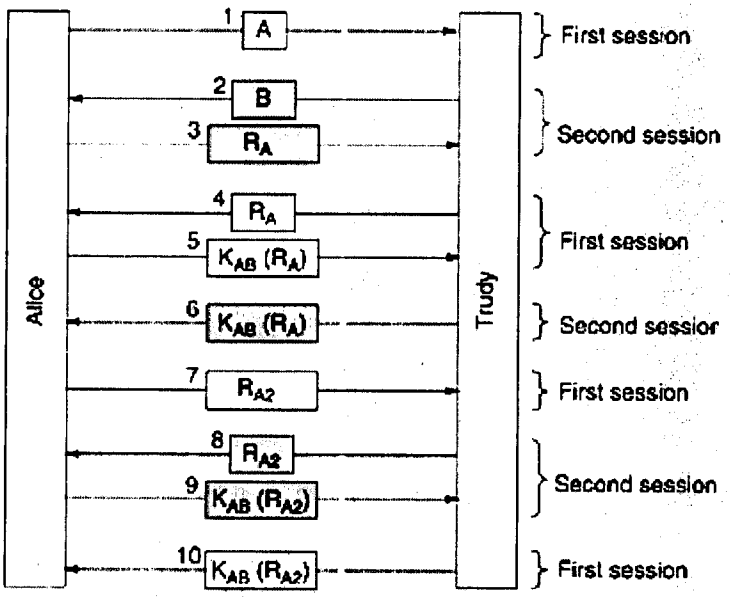


Figure 5.35. A reflection attack on the protocol of Figure 5.32.

What does Trudy do now? She goes back to the first session, where it is her turn to send a challenge, and sends the R_A she got in message 3. Alice kindly responds to it in message 5, thus supplying Trudy with the information she needs to send message 6 in session 2. At this point, Trudy is basically home free because she has successfully responded to Alice's challenge in session 2. She can now cancel session 1, send over any old number for the rest of session 2, and she will have an authenticated session with Alice in session 2.

But Trudy is nasty, and she really wants to rub it in. Instead of sending any old number over to complete session 2, she waits until Alice sends message 7, Alice's challenge for session 1. Of course, Trudy does not know how to respond, so she uses the reflection attack again, sending back R_{A2} as message 8. Alice conveniently encrypts R_{A2} in message 9. Trudy now switches back to session 1 and sends Alice the number she wants in message 10, conveniently copied from what Alice sent in message 9. At this point Trudy has two fully authenticated sessions with Alice.

This attack has a somewhat different result than the attack on the three-message protocol shown in Figure 5.34. This time, Trudy has two authenticated connections with Alice. In the previous example, she had one authenticated connection with Bob. Again here, if we had applied all the general authentication protocol rules discussed above, this attack could have been stopped. A detailed discussion of these kind of attacks and how to thwart them is given in. They also show how it is possible to systematically construct protocols that are probably correct. The simplest such protocol is nevertheless a bit complicated, so we will now show a different class of protocol that also works.

The new authentication protocol is shown in Figure 5.36. It uses an HMAC of the type we saw when studying IPsec. Alice starts out by sending Bob a nonce, R_A as message 1. Bob responds by selecting his own nonce, R_B, and sending it back along with an HMAC. The HMAC is formed to building a data structure consisting of the Alice's nonce, Bob's nonce, their identities, and the shared secret key, K_{AB}. This data structured is then hashed into the HMAC, for example using SHA-1. When Alice receives message 2, she now has R_A (which she picked herself), R_B, which arrives as plaintext, the two identities, and the secret key, K_{AB}, which has known all along, so she can compute the HMAC herself. If it agrees with the HMAC in the message, she knows she is talking to Bob because Trudy does not know K_{AB} and thus cannot figure out which HMAC to send. Alice responds to Bob with an HMAC containing just the two nonces.

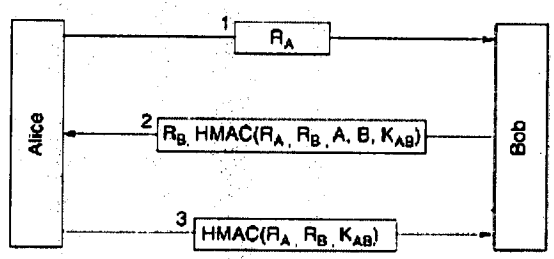


Figure 5.36. Authentication using HMACs.

Can Trudy somehow subvert this protocol? No, because she cannot force either party to encrypt or hash a value of her choice, as happened in Figure 5.34 and Figure 5.36. Both HMACs include values chosen by the sending party, something which Trudy cannot control.

Using HMACs is not the only way to use this idea. An alternative scheme that is often used instead of computing the HMAC over a series of items is to encrypt the items sequentially using cipher block chaining.

Establishing a Shared Key: The Diffie-Hellman Key Exchange

So far we have assumed that Alice and Bob share a secret key. Suppose that they do not (because so far there is no universally accepted PKI for signing and distributing certificates). How can they establish one? One way would be for Alice to call Bob and give him her key on the phone, but he would probably start out by saying: How do I know you are Alice and not Trudy? They could try to arrange a meeting, with each one bringing a passport, a drivers' license, and three major credit cards, but being busy people, they might not be able to find a mutually acceptable date for months. Fortunately, incredible as it may sound, there is a way for total strangers to establish a shared secret key in broad daylight, even with Trudy carefully recording every message.

The protocol that allows strangers to establish a shared secret key is called the Diffie-Hellman key exchange and works as follows. Alice and Bob have to agree on two large numbers, n and g, where n is a prime, (n - 1)/2 is also a prime and certain conditions apply to g. These numbers may be public, so either one of them can just pick n and g and tell the other openly. Now Alice picks a large (say, 512-bit) number, x, and keeps it secret. Similarly, Bob picks a large secret number, y.

Alice initiates the key exchange protocol by sending Bob a message containing (n, g, g^x mod n), as shown in Figure 5.37. Bob responds by sending Alice a message containing g^y mod n. Now Alice raises the number Bob sent her to the xth power modulo n to get $(g^y \bmod n)^x \bmod n$. Bob performs a similar operation to get $(g^x \bmod n)^y \bmod n$. By the laws of modular arithmetic, both calculations yield $g^{xy} \bmod n$. Lo and behold, Alice and Bob suddenly share a secret key, $g^{xy} \bmod n$.

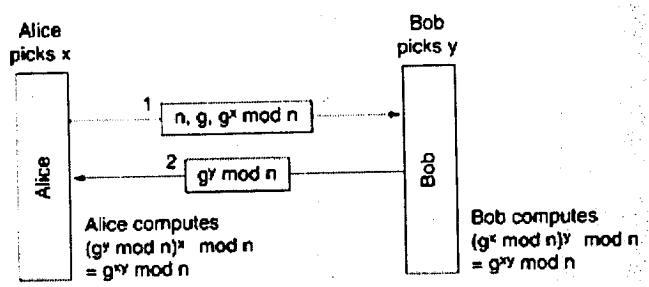


Figure 5.37. The Diffie-Hellman key exchange.

Trudy, of course, has seen both messages. She knows g and n from message 1. If she could compute x and y, she could figure out the secret key. The trouble is, given only $g^x \bmod n$, she cannot find x. No practical

algorithm for computing discrete logarithms modulo a very large prime number is known.

To make the above example more concrete, we will use the (completely unrealistic) values of n = 47 and g = 3. Alice picks x = 8 and Bob picks y = 10. Both of these are kept secret. Alice's message to Bob is (47, 3, 28) because 3^8 mod 47 is 28. Bob's message to Alice is (17). Alice computes 17^8 mod 47, which is 4. Bob computes 28^{10} mod 47, which is 4. Alice and Bob have independently determined that the secret key is now 4. Trudy has to solve the equation 3^x mod 47 = 28, which can be done by exhaustive search for small numbers like this, but not when all the numbers are hundreds of bits long. All currently-known algorithms simply take too long, even on massively parallel supercomputers.

Despite the elegance of the Diffie-Hellman algorithm, there is a problem: when Bob gets the triple (47, 3, 28), how does he know it is from Alice and not from Trudy? There is no way he can know. Unfortunately, Trudy can exploit this fact to deceive both Alice and Bob, as illustrated in Figure 5.38. Here, while Alice and Bob are choosing x and y, respectively, Trudy picks her own random number, z. Alice sends message 1 intended for Bob. Trudy intercepts it and sends message 2 to Bob, using the correct g and n (which are public anyway) but with her own z instead of x. She also sends message 3 back to Alice. Later Bob sends message 4 to Alice, which Trudy again intercepts and keeps.

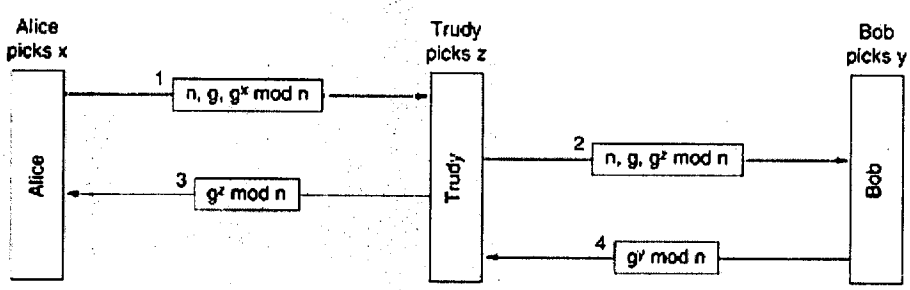


Figure 5.38. The bucket brigade or man-in-the-middle attack.

Now everybody does the modular arithmetic. Alice computes the secret key as g^{xz} mod n, and so does Trudy (for messages to Alice). Bob computes g^{yz} mod n and so does Trudy (for messages to Bob). Alice thinks she is talking to Bob so she establishes a session key (with Trudy). So does Bob. Every message that Alice sends on the encrypted session is captured by Trudy, stored, modified if desired, and then (optionally) passed on to Bob.

Similarly, in the other direction. Trudy sees everything and can modify all messages at will, while both Alice and Bob are under the illusion that they have a secure channel to one another. This attack is known as the bucket brigade attack, because it vaguely resembles an old-time volunteer fire department passing buckets along the line from the fire truck to the fire. It is also called the man-in-the-middle attack.

Authentication Using a Key Distribution Center

Setting up a shared secret with a stranger almost worked, but not quite. On the other hand, it probably was not worth doing in the first place (sour grapes attack). To talk to n people this way, you would need n keys. For popular people, key management would become a real burden, especially if each key had to be stored on a separate plastic chip card.

A different approach is to introduce a trusted key distribution center (KDC). In this model, each user has a single key shared with the KDC. Authentication and session key management now goes through the KDC. The simplest known KDC authentication protocol involving two parties and a trusted KDC is depicted in Figure 5.39.

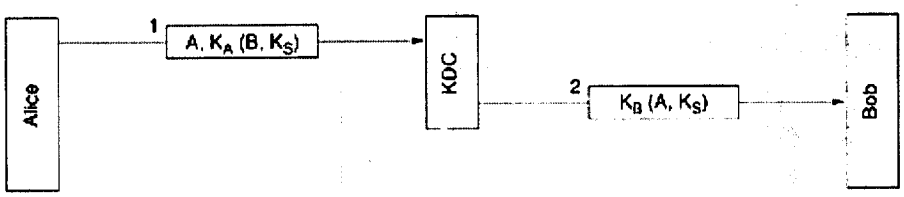


Figure 5.39. A first attempt at an authentication protocol using a KDC.

The idea behind this protocol is simple: Alice picks a session key, K_S, and tells the KDC that she wants to talk to Bob using K_S. This message is encrypted with the secret key Alice shares (only) with the KDC, K_A. The KDC decrypts this message, extracting Bob's identity and the session key. It then constructs a new message containing Alice's identity and the session key and sends this message to Bob. This encryption is done with K_B, the secret key Bob shares with the KDC. When Bob decrypts the message, he learns that Alice wants to talk to him and which key she wants to use.

The authentication here happens for free. The KDC knows that message 1 must have come from Alice, since no one else would have been able to encrypt it with Alice's secret key. Similarly, Bob knows that message 2 must

have come from the KDC, whom he trusts, since no one else knows his secret key.

Unfortunately, this protocol has a serious flaw. Trudy needs some money, so she figures out some legitimate service she can perform for Alice, makes an attractive offer, and gets the job. After doing the work, Trudy then politely requests Alice to pay by bank transfer. Alice then establishes a session key with her banker, Bob. Then she sends Bob a message requesting money to be transferred to Trudy's account.

Meanwhile, Trudy is back to her old ways, snooping on the network. She copies both message 2 in Figure 5.39 and the money-transfer request that follows it. Later, she replays both of them to Bob. Bob gets them and thinks: Alice must have hired Trudy again. She clearly does good work. Bob then transfers an equal amount of money from Alice's account to Trudy's. Sometime after the 50th message pair, Bob runs out of the office to find Trudy to offer her a big loan so she can expand her obviously successful business. This problem is called the replay attack.

Several solutions to the replay attack are possible. The first one is to include a timestamp in each message. Then if anyone receives an obsolete message, it can be discarded. The trouble with this approach is that clocks are never exactly synchronized over a network, so there has to be some interval during which a timestamp is valid. Trudy can replay the message during this interval and get away with it.

The second solution is to put a nonce in each message. Each party then has to remember all previous nonces and reject any message containing a previously-used nonce. But nonces have to be remembered forever, lest Trudy try replaying a 5-year-old message. Also, if some machine crashes and it loses its nonce list, it is again vulnerable to a replay attack. Timestamps and nonces can be combined to limit how long nonces have to be remembered, but clearly the protocol is going to get a lot more complicated.

A more sophisticated approach to mutual authentication is to use a multiway challenge-response protocol. A well-known example of such a protocol is the Needham-Schroeder authentication protocol, one variant of which is shown in Figure 5.40.

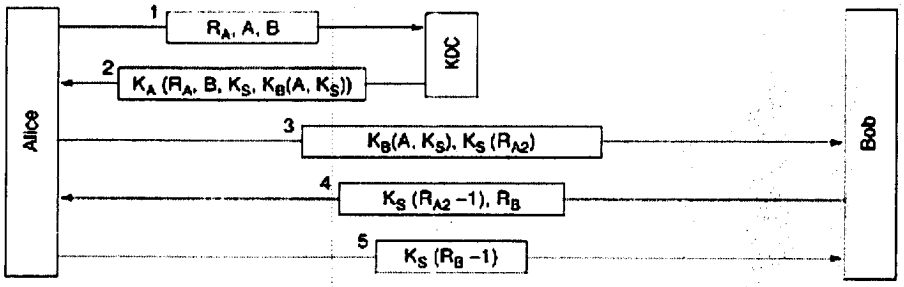


Figure 5.40. The Needham-Schroeder authentication protocol.

The protocol begins with Alice telling the KDC that she wants to talk to Bob. This message contains a large random number, R_A, as a nonce. The KDC sends back message 2 containing Alice's random number, a session key, and a ticket that she can send to Bob. The point of the random number, R_A, is to assure Alice that message 2 is fresh, and not a replay. Bob's identity is also enclosed in case Trudy gets any funny ideas about replacing B in message 1 with her own identity so the KDC will encrypt the ticket at the end of message 2 with K_T instead of K_B. The ticket encrypted with K_B is included inside the encrypted message to prevent Trudy from replacing it with something else on the way back to Alice.

Alice now sends the ticket to Bob, along with a new random number, R_{A2}, encrypted with the session key, K_S. In message 4, Bob sends back $K_S(R_{A2} - 1)$ to prove to Alice that she is talking to the real Bob. Sending back $K_S(R_{A2})$ would not have worked, since Trudy could just have stolen it from message 3.

After receiving message 4, Alice is now convinced that she is talking to Bob and that no replays could have been used so far. After all, she just generated R_{A2} a few milliseconds ago. The purpose of message 5 is to convince Bob that it is indeed Alice he is talking to, and no replays are being used here either. By having each party both generate a challenge and respond to one, the possibility of any kind of replay attack is eliminated.

Although this protocol seems pretty solid, it does have a slight weakness. If Trudy ever manages to obtain an old session key in plaintext, she can initiate a new session with Bob by replaying the message 3 corresponding to the compromised key and convince him that she is Alice. This time she can plunder Alice's bank account without having to perform the legitimate service even once.

Needham and Schroeder later published a protocol that corrects this problem (Needham and Schroeder, 1987). In the same issue of the same journal, Otway and Rees also published a protocol that solves the problem in a shorter way. Figure 5.41 shows a slightly modified Otway-Rees protocol.

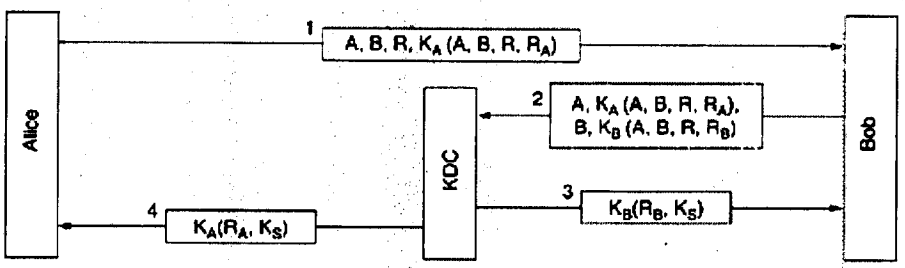


Figure 5.41. The Otway-Rees authentication protocol (slightly simplified).

In the Otway-Rees protocol, Alice starts out by generating a pair of random numbers, R, which will be used as a common identifier, and R_A, which Alice will use to challenge Bob. When Bob gets this message, he constructs a new message from the encrypted part of Alice's message and an analogous one of his own. Both the parts encrypted with K_A and K_B identify Alice and Bob, contain the common identifier, and contain a challenge.

The KDC checks to see if the R in both parts is the same. It might not be because Trudy tampered with R in message 1 or replaced part of message 2. If the two Rs match, the KDC believes that the request message from Bob is valid. It then generates a session key and encrypts it twice, once for Alice and once for Bob. Each message contains the receiver's random number, as proof that the KDC, and not Trudy, generated the message. At this point both Alice and Bob are in possession of the same session key and can start communicating. The first time they exchange data messages, each one can see that the other one has an identical copy of K_S, so the authentication is then complete.

Authentication Using Kerberos

An authentication protocol used in many real systems is Kerberos, which is based on a variant of Needham-Schroeder. It is named for a multiheaded dog in Greek mythology that used to guard the entrance to Hades (presumably to keep undesirables out). Kerberos was designed at M.I.T. to allow workstation users to access network resources in a secure way. Its biggest difference from Needham-Schroeder is its assumption that all clocks are fairly well synchronized. The protocol has gone through several

iterations. V4 is the version most widely used in industry, so we will describe it. Afterward, we will say a few words about its successor, V5.

Kerberos involves three servers in addition to Alice (a client workstation):

Authentication Server (AS): verifies users during login

Ticket-Granting Server (TGS): issues "proof of identity tickets"

Bob the server: actually does the work Alice wants performed

AS is similar to a KDC in that it shares a secret password with every user. The TGS's job is to issue tickets that can convince the real servers that the bearer of a TGS ticket really is who he or she claims to be.

To start a session, Alice sits down at an arbitrary public workstation and types her name. The workstation sends her name to the AS in plaintext, as shown in Figure 5.42. What comes back is a session key and a ticket, $K_{TGS}(A, K_S)$, intended for the TGS. These items are packaged together and encrypted using Alice's secret key, so that only Alice can decrypt them. Only when message 2 arrives does the workstation ask for Alice's password. The password is then used to generate K_A in order to decrypt message 2 and obtain the session key and TGS ticket inside it. At this point, the workstation overwrites Alice's password to make sure that it is only inside the workstation for a few milliseconds at most. If Trudy tries logging in as Alice, the password she types will be wrong and the workstation will detect this because the standard part of message 2 will be incorrect.

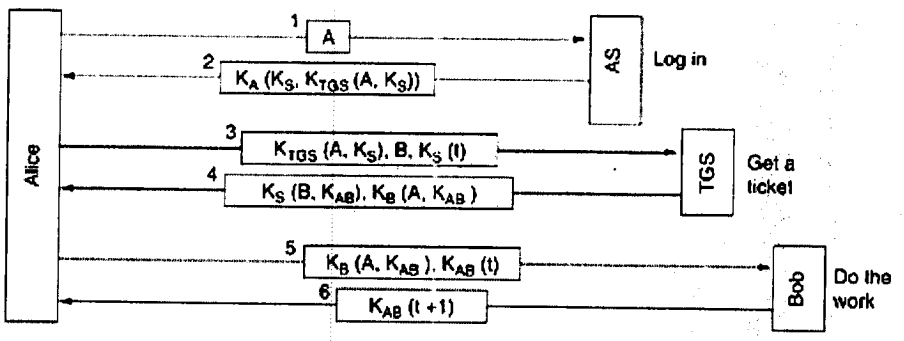


Figure 5.42. The operation of Kerberos V4.

After she logs in, Alice may tell the workstation that she wants to contact Bob the file server. The workstation then sends message 3 to the TGS asking for a ticket to use with Bob. The key element in this request is $K_{TGS}(A, K_S)$, which is encrypted with the TGS's secret key and used as proof that the sender really is Alice. The TGS responds by creating a session key, K_{AB}, for Alice to use with Bob. Two versions of it are sent

back. The first is encrypted with only K_S, so Alice can read it. The second is encrypted with Bob's key, K_B, so Bob can read it.

Trudy can copy message 3 and try to use it again, but she will be foiled by the encrypted timestamp, t, sent along with it. Trudy cannot replace the timestamp with a more recent one, because she does not know K_S, the session key Alice uses to talk to the TGS. Even if Trudy replays message 3 quickly, all she will get is another copy of message 4, which she could not decrypt the first time and will not be able to decrypt the second time either.

Now Alice can send K_{AB} to Bob to establish a session with him. This exchange is also timestamped. The response is proof to Alice that she is actually talking to Bob, not to Trudy.

After this series of exchanges, Alice can communicate with Bob under cover of K_{AB}. If she later decides she needs to talk to another server, Carol, she just repeats message 3 to the TGS, only now specifying C instead of B. The TGS will promptly respond with a ticket encrypted with K_C that Alice can send to Carol and that Carol will accept as proof that it came from Alice.

The point of all this work is that now Alice can access servers all over the network in a secure way and her password never has to go over the network. In fact, it only had to be in her own workstation for a few milliseconds. However, note that each server does its own authorization. When Alice presents her ticket to Bob, this merely proves to Bob who sent it. Precisely what Alice is allowed to do is up to Bob.

Since the Kerberos designers did not expect the entire world to trust a single authentication server, they made provision for having multiple realms, each with its own AS and TGS. To get a ticket for a server in a distant realm, Alice would ask her own TGS for a ticket accepted by the TGS in the distant realm. If the distant TGS has registered with the local TGS (the same way local servers do), the local TGS will give Alice a ticket valid at the distant TGS. She can then do business over there, such as getting tickets for servers in that realm. Note, however, that for parties in two realms to do business, each one must trust the other's TGS.

Kerberos V5 is fancier than V4 and has more overhead. It also uses OSI ASN.1 (Abstract Syntax Notation 1) for describing data types and has small changes in the protocols. Furthermore, it has longer ticket lifetimes, allows

tickets to be renewed, and will issue postdated tickets. In addition, at least in theory, it is not DES dependent, as V4 is, and supports multiple realms by delegating ticket generation to multiple ticket servers.

Authentication Using Public-Key Cryptography

Mutual authentication can also be done using public-key cryptography. To start with, Alice needs to get Bob's public key. If a PKI exists with a directory server that hands out certificates for public keys, Alice can ask for Bob's, as shown in Figure 5.43 as message 1. The reply, in message 2, is an X.509 certificate containing Bob's public key. When Alice verifies that the signature is correct, she sends Bob a message containing her identity and a nonce.

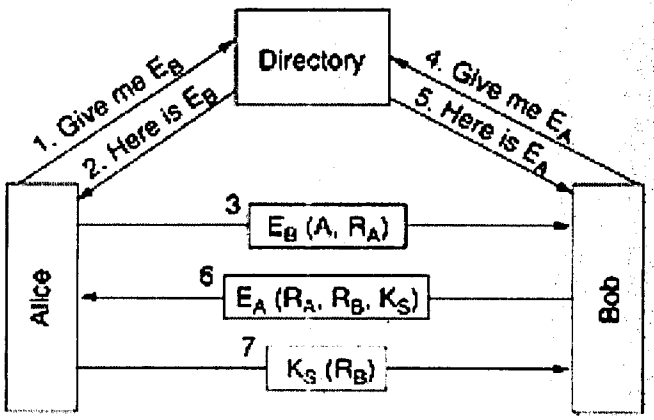


Figure 5.43. Mutual authentication using public-key cryptography.

When Bob receives this message, he has no idea whether it came from Alice or from Trudy, but he plays along and asks the directory server for Alice's public key (message 4) which he soon gets (message 5). He then sends Alice a message containing Alice's R_A, his own nonce, R_B, and a proposed session key, K_S, as message 6.

When Alice gets message 6, she decrypts it using her private key. She sees R_A in it, which gives her a warm feeling inside. The message must have come from Bob, since Trudy has no way of determining R_A. Furthermore, it must be fresh and not a replay, since she just sent Bob R_A. Alice agrees to the session by sending back message 7. When Bob sees R_B encrypted with the session key he just generated, he knows Alice got message 6 and verified R_A.

What can Trudy do to try to subvert this protocol? She can fabricate message 3 and trick Bob into probing Alice, but Alice will see an R_A that she did not send and will not proceed further. Trudy cannot forge message 7 back to Bob because she does not know R_B or K_S and cannot determine them without Alice's private key. She is out of luck.

5.8 E-Mail Security

When an e-mail message is sent between two distant sites, it will generally transit dozens of machines on the way. Any of these can read and record the message for future use. In practice, privacy is nonexistent, despite what many people think. Nevertheless, many people would like to be able to send e-mail that can be read by the intended recipient and no one else: not their boss and not even their government. This desire has stimulated several people and groups to apply the cryptographic principles we studied earlier to e-mail to produce secure e-mail. In the following sections we will study a widely-used secure e-mail system, PGP, and then briefly mention two others, PEM and S/MIME.

PGP—Pretty Good Privacy

Our first example, PGP (Pretty Good Privacy) is essentially the brainchild of one person, Phil Zimmermann. Zimmermann is a privacy advocate whose motto is: If privacy is outlawed, only outlaws will have privacy. Released in 1991, PGP is a complete e-mail security package that provides privacy, authentication, digital signatures, and compression, all in an easy-to-use form. Furthermore, the complete package, including all the source code, is distributed free of charge via the Internet. Due to its quality, price (zero), and easy availability on UNIX, Linux, Windows, and Mac OS platforms, it is widely used today.

PGP encrypts data by using a block cipher called IDEA (International Data Encryption Algorithm), which uses 128-bit keys. It was devised in Switzerland at a time when DES was seen as tainted and AES had not yet been invented. Conceptually, IDEA is similar to DES and AES: it mixes up the bits in a series of rounds, but the details of the mixing functions are different from DES and AES. Key management uses RSA and data integrity uses MD5, topics that we have already discussed.

PGP has also been embroiled in controversy since day 1. Because Zimmermann did nothing to stop other people from placing PGP on the

Internet, where people all over the world could get it, the U.S. Government claimed that Zimmermann had violated U.S. laws prohibiting the export of munitions. The U.S. Government's investigation of Zimmermann went on for 5 years, but was eventually dropped, probably for two reasons. First, Zimmermann did not place PGP on the Internet himself, so his lawyer claimed that he never exported anything (and then there is the little matter of whether creating a Web site constitutes export at all). Second, the government eventually came to realize that winning a trial meant convincing a jury that a Web site containing a downloadable privacy program was covered by the arms-trafficking law prohibiting the export of war materiel such as tanks, submarines, military aircraft, and nuclear weapons. Years of negative publicity probably did not help much, either.

As an aside, the export rules are bizarre, to put it mildly. The government considered putting code on a Web site to be an illegal export and harassed Zimmermann for 5 years about it. On the other hand, when someone published the complete PGP source code, in C, as a book (in a large font with a checksum on each page to make scanning it in easy) and then exported the book, that was fine with the government because books are not classified as munitions. The sword is mightier than the pen, at least for Uncle Sam.

Another problem PGP ran into involved patent infringement. The company holding the RSA patent, RSA Security, Inc., alleged that PGP's use of the RSA algorithm infringed on its patent, but that problem was settled with releases starting at 2.6. Furthermore, PGP uses another patented encryption algorithm, IDEA, whose use caused some problems at first.

Since PGP is open source, various people and groups have modified it and produced a number of versions. Some of these were designed to get around the munitions laws, others were focused on avoiding the use of patented algorithms, and still others wanted to turn it into a closed-source commercial product. Although the munitions laws have now been slightly liberalized (otherwise products using AES would not have been exportable from the U.S.), and the RSA patent expired in September 2000, the legacy of all these problems is that several incompatible versions of PGP are in circulation, under various names. The discussion below focuses on classic PGP, which is the oldest and simplest version. Another popular version, Open PGP, is described in RFC 2440. Yet another is the GNU Privacy Guard.

PGP intentionally uses existing cryptographic algorithms rather than inventing new ones. It is largely based on algorithms that have withstood extensive peer review and were not designed or influenced by any government agency trying to weaken them. For people who tend to distrust government, this property is a big plus.

PGP supports text compression, secrecy, and digital signatures and also provides extensive key management facilities, but oddly enough, not e-mail facilities. It is more of a preprocessor that takes plaintext as input and produces signed ciphertext in base64 as output. This output can then be e-mailed, of course. Some PGP implementations call a user agent as the final step to actually send the message.

To see how PGP works, let us consider the example of Figure 5.44. Here, Alice wants to send a signed plaintext message, P, to Bob in a secure way. Both Alice and Bob have private (D_X) and public (E_X) RSA keys. Let us assume that each one knows the other's public key; we will cover PGP key management shortly.

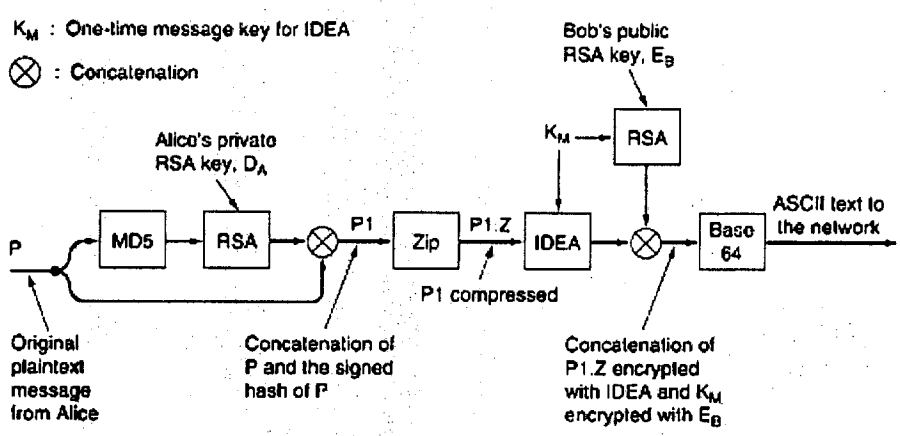


Figure 5.44. PGP in operation for sending a message.

Alice starts out by invoking the PGP program on her computer. PGP first hashes her message, P, using MD5, and then encrypts the resulting hash using her private RSA key, D_A. When Bob eventually gets the message, he can decrypt the hash with Alice's public key and verify that the hash is correct. Even if someone else (e.g., Trudy) could acquire the hash at this stage and decrypt it with Alice's known public key, the strength of MD5 guarantees that it would be computationally infeasible to produce another message with the same MD5 hash.

The encrypted hash and the original message are now concatenated into a single message, P1, and compressed using the ZIP program, which uses the Ziv-Lempel algorithm. Call the output of this step P1.Z.

Next, PGP prompts Alice for some random input. Both the content and the typing speed are used to generate a 128-bit IDEA message key, K_M (called a session key in the PGP literature, but this is really a misnomer since there is no session). K_M is now used to encrypt P1.Z with IDEA in cipher feedback mode. In addition, K_M is encrypted with Bob's public key, E_B. These two components are then concatenated and converted to base64. The resulting message then contains only letters, digits, and the symbols +, /, and =, which means it can be put into an RFC 822 body and be expected to arrive unmodified.

When Bob gets the message, he reverses the base64 encoding and decrypts the IDEA key using his private RSA key. Using this key, he decrypts the message to get P1.Z. After decompressing it, Bob separates the plaintext from the encrypted hash and decrypts the hash using Alice's public key. If the plaintext hash agrees with his own MD5 computation, he knows that P is the correct message and that it came from Alice.

It is worth noting that RSA is only used in two places here: to encrypt the 128-bit MD5 hash and to encrypt the 128-bit IDEA key. Although RSA is slow, it has to encrypt only 256 bits, not a large volume of data. Furthermore, all 256 plaintext bits are exceedingly random, so a considerable amount of work will be required on Trudy's part just to determine if a guessed key is correct. The heavy duty encryption is done by IDEA, which is orders of magnitude faster than RSA. Thus, PGP provides security, compression, and a digital signature and does so in a much more efficient way than the scheme illustrated in Figure 5.19.

PGP supports four RSA key lengths. It is up to the user to select the one that is most appropriate. The lengths are

1. Casual (384 bits): can be broken easily today.
2. Commercial (512 bits): breakable by three-letter organizations.
3. Military (1024 bits): Not breakable by anyone on earth.
4. Alien (2048 bits): Not breakable by anyone on other planets, either.

Since RSA is only used for two small computations, everyone should use alien strength keys all the time.

The format of a classic PGP message is shown in Figure 5.45. Numerous other formats are also in use. The message has three parts, containing the IDEA key, the signature, and the message, respectively. The key part contains not only the key, but also a key identifier, since users are permitted to have multiple public keys.

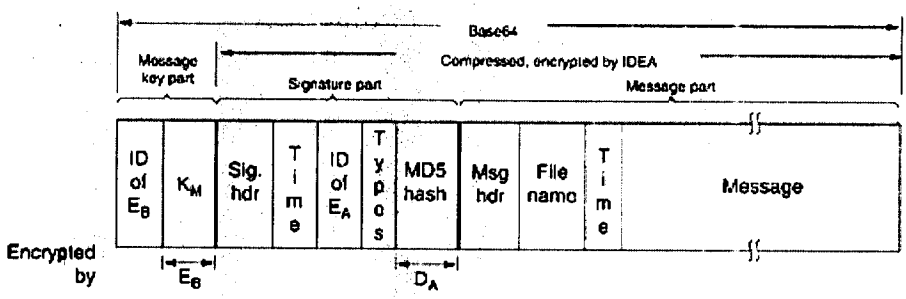


Figure 5.45. A PGP message.

The signature part contains a header, which will not concern us here. The header is followed by a timestamp, the identifier for the sender's public key that can be used to decrypt the signature hash, some type information that identifies the algorithms used (to allow MD6 and RSA2 to be used when they are invented), and the encrypted hash itself.

The message part also contains a header, the default name of the file to be used if the receiver writes the file to the disk, a message creation timestamp, and, finally, the message itself.

Key management has received a large amount of attention in PGP as it is the Achilles heel of all security systems. Key management works as follows. Each user maintains two data structures locally: a private key ring and a public key ring. The private key ring contains one or more personal private-public key pairs. The reason for supporting multiple pairs per user is to permit users to change their public keys periodically or when one is thought to have been compromised, without invalidating messages currently in preparation or in transit. Each pair has an identifier associated with it so that a message sender can tell the recipient which public key was used to encrypt it. Message identifiers consist of the low-order 64 bits of the public key. Users are responsible for avoiding conflicts in their public key

identifiers. The private keys on disk are encrypted using a special (arbitrarily long) password to protect them against sneak attacks.

The public key ring contains public keys of the user's correspondents. These are needed to encrypt the message keys associated with each message. Each entry on the public key ring contains not only the public key, but also its 64-bit identifier and an indication of how strongly the user trusts the key.

The problem being tackled here is the following. Suppose that public keys are maintained on bulletin boards. One way for Trudy to read Bob's secret e-mail is to attack the bulletin board and replace Bob's public key with one of her choice. When Alice later fetches the key allegedly belonging to Bob, Trudy can mount a bucket brigade attack on Bob.

To prevent such attacks, or at least minimize the consequences of them, Alice needs to know how much to trust the item called "Bob's key" on her public key ring. If she knows that Bob personally handed her a floppy disk containing the key, she can set the trust value to the highest value. It is this decentralized, user-controlled approach to public-key management that sets PGP apart from centralized PKI schemes.

Nevertheless, people do sometimes obtain public keys by querying a trusted key server. For this reason, after X.509 was standardized, PGP supported these certificates as well as the traditional PGP public key ring mechanism. All current versions of PGP have X.509 support.

PEM—Privacy Enhanced Mail

In contrast to PGP, which was initially a one-man show, our second example, PEM (Privacy Enhanced Mail), developed in the late 1980s, is an official Internet standard and described in four RFCs: RFC 1421 through RFC 1424. Very roughly, PEM covers the same territory as PGP: privacy and authentication for RFC 822-based e-mail systems. Nevertheless, it also has some differences from PGP in approach and technology.

Messages sent using PEM are first converted to a canonical form so they all have the same conventions about white space (e.g., tabs, trailing spaces). Next, a message hash is computed using MD2 or MD5. Then the concatenation of the hash and the message is encrypted using DES. In light of the known weakness of a 56-bit key, this choice is certainly suspect.

The encrypted message can then be encoded with base64 coding and transmitted to the recipient.

As in PGP, each message is encrypted with a one-time key that is enclosed along with the message. The key can be protected either with RSA or with triple DES using EDE.

Key management is more structured than in PGP. Keys are certified by X.509 certificates issued by CAs, which are arranged in a rigid hierarchy starting at a single root. The advantage of this scheme is that certificate revocation is possible by having the root issue CRLs periodically.

The only problem with PEM is that nobody ever used it and it has long-since gone to that big bit bin in the sky. The problem was largely political: who would operate the root and under what conditions? There was no shortage of candidates, but many people were afraid to trust anyone company with the security of the whole system. The most serious candidate, RSA Security, Inc., wanted to charge per certificate issued. However, some organizations balked at this idea. In particular, the U.S. Government is allowed to use all U.S. patents for free, and companies outside the U.S. had become accustomed to using the RSA algorithm for free (the company forgot to patent it outside the U.S.). Neither was enthusiastic about suddenly having to pay RSA Security, Inc., for doing something that they had always done for free. In the end, no root could be found and PEM collapsed.

S/MIME

IETF's next venture into e-mail security, called S/MIME (Secure/MIME), is described in RFCs 2632 through 2643. Like PEM, it provides authentication, data integrity, secrecy, and nonrepudiation. It also is quite flexible, supporting a variety of cryptographic algorithms. Not surprisingly, given the name, S/MIME integrates well with MIME, allowing all kinds of messages to be protected. A variety of new MIME headers are defined, for example, for holding digital signatures.

IETF definitely learned something from the PEM experience. S/MIME does not have a rigid certificate hierarchy beginning at a single root. Instead, users can have multiple trust anchors. As long as a certificate can be traced back to some trust anchor the user believes in, it is considered valid. S/MIME uses the standard algorithms and protocols we have been

examining so far, so we will not discuss it any further here. For the details, please consult the RFCs.

5.9 Web Security

We have just studied two important areas where security is needed: communications and e-mail. You can think of these as the soup and appetizer. Now it is time for the main course: Web security. The Web is where most of the Trudies hang out nowadays and do their dirty work. In the following sections we will look at some of the problems and issues relating to Web security.

Web security can be roughly divided into three parts. First, how are objects and resources named securely? Second, how can secure, authenticated connections be established? Third, what happens when a Web site sends a client a piece of executable code? After looking at some threats, we will examine all these issues.

Threats

One reads about Web site security problems in the newspaper almost weekly. The situation is really pretty grim. Let us look at a few examples of what has already happened. First, the home page of numerous organizations has been attacked and replaced by a new home page of the crackers' choosing. (The popular press calls people who break into computers "hackers," but many programmers reserve that term for great programmers. We prefer to call these people "crackers.") Sites that have been cracked include Yahoo, the U.S. Army, the CIA, NASA, and the New York Times. In most cases, the crackers just put up some funny text and the sites were repaired within a few hours.

Now let us look at some much more serious cases. Numerous sites have been brought down by denial-of-service attacks, in which the cracker floods the site with traffic, rendering it unable to respond to legitimate queries. Often the attack is mounted from a large number of machines that the cracker has already broken into (DDoS atacks). These attacks are so common that they do not even make the news any more, but they can cost the attacked site thousands of dollars in lost business.

In 1999, a Swedish cracker broke into Microsoft's Hotmail Web site and created a mirror site that allowed anyone to type in the name of a Hotmail user and then read all of the person's current and archived e-mail.

In another case, a 19-year-old Russian cracker named Maxim broke into an e-commerce Web site and stole 300,000 credit card numbers. Then he approached the site owners and told them that if they did not pay him $100,000, he would post all the credit card numbers to the Internet. They did not give in to his blackmail, and he indeed posted the credit card numbers, inflicting great damage to many innocent victims.

In a different vein, a 23-year-old California student e-mailed a press release to a news agency falsely stating that the Emulex Corporation was going to post a large quarterly loss and that the C.E.O. was resigning immediately. Within hours, the company's stock dropped by 60%, causing stockholders to lose over $2 billion. The perpetrator made a quarter of a million dollars by selling the stock short just before sending the announcement. While this event was not a Web site break-in, it is clear that putting such an announcement on the home page of any big corporation would have a similar effect.

We could (unfortunately) go on like this for many pages. But it is now time to examine some of the technical issues related to Web security. For more information about security problems of all kinds. Searching the Internet will also turn up vast numbers of specific cases.

Secure Naming

Let us start with something very basic: Alice wants to visit Bob's Web site. She types Bob's URL into her browser and a few seconds later, a Web page appears. But is it Bob's? May be yes and maybe no. Trudy might be up to her old tricks again. For example, she might be intercepting all of Alice's outgoing packets and examining them. When she captures an HTTP GET request headed to Bob's Web site, she could go to Bob's Web site herself to get the page, modify it as she wishes, and return the fake page to Alice. Alice would be none the wiser. Worse yet, Trudy could slash the prices at Bob's e-store to make his goods look very attractive, thereby tricking Alice into sending her credit card number to "Bob" to buy some merchandise.

One disadvantage to this classic man-in-the-middle attack is that Trudy has to be in a position to intercept Alice's outgoing traffic and forge her incoming traffic. In practice, she has to tap either Alice's phone line or Bob's, since tapping the fiber backbone is fairly difficult. While active wiretapping is certainly possible, it is a certain amount of work, and while

Trudy is clever, she is also lazy. Besides, there are easier ways to trick Alice.

DNS Spoofing

For example, suppose Trudy is able to crack the DNS system, maybe just the DNS cache at Alice's ISP, and replace Bob's IP address (say, 36.1.2.3) with her (Trudy's) IP address (say, 42.9.9.9). That leads to the following attack. The way it is supposed to work is illustrated in Figure 5.46. Here (1) Alice asks DNS for Bob's IP address, (2) gets it, (3) asks Bob for his home page, and (4) gets that, too. After Trudy has modified Bob's DNS record to contain her own IP address instead of Bob's, we get the situation of Figure 5.46(b). Here, when Alice looks up Bob's IP address, she gets Trudy's, so all her traffic intended for Bob goes to Trudy. Trudy can now mount a man-in-the-middle attack without having to go to the trouble of tapping any phone lines. Instead, she has to break into a DNS server and change one record, a much easier proposition.

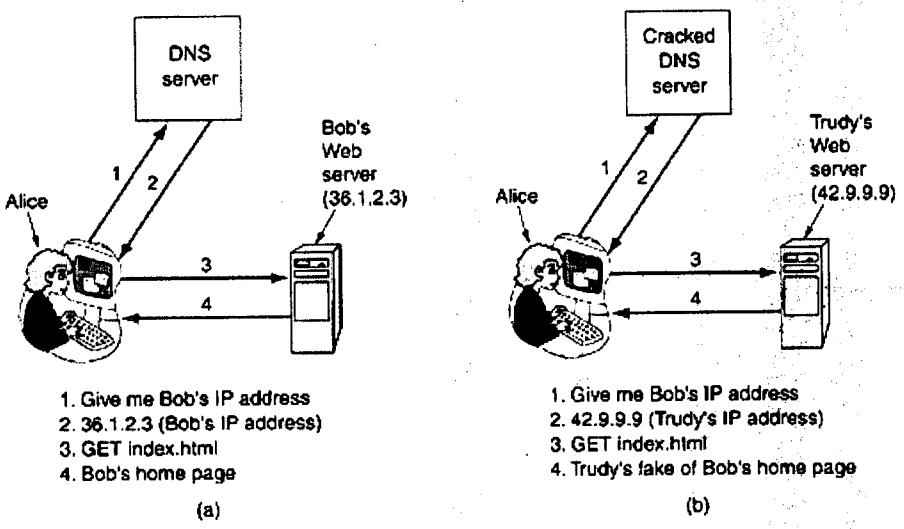


Figure 5.46. (a) Normal situation. (b) An attack based on breaking into DNS and modifying Bob's record.

How might Trudy fool DNS? It turns out to be relatively easy. Briefly summarized, Trudy can trick the DNS server at Alice's ISP into sending out a query to look up Bob's address. Unfortunately, since DNS uses UDP, the DNS server has no real way of checking who supplied the answer. Trudy can exploit this property by forging the expected reply and thus injecting a

false IP address into the DNS server's cache. For simplicity, we will assume that Alice's ISP does not initially have an entry for Bob's Web site, bob.com. If it does, Trudy can wait until it times out and try later (or use other tricks).

Trudy starts the attack by sending a lookup request to Alice's ISP asking for the IP address of bob.com. Since there is no entry for this DNS name, the cache server queries the top-level server for the com domain to get one. However, Trudy beats the com server to the punch and sends back a false reply saying: "bob.com is 42.9.9.9," where that IP address is hers. If her false reply gets back to Alice's ISP first, that one will be cached and the real reply will be rejected as an unsolicited reply to a query no longer outstanding. Tricking a DNS server into installing a false IP address is called DNS spoofing. A cache that holds an intentionally false IP address like this is called a poisoned cache.

Actually, things are not quite that simple. First, Alice's ISP checks to see that the reply bears the correct IP source address of the top-level server. But since Trudy can put anything she wants in that IP field, she can defeat that test easily since the IP addresses of the top-level servers have to be public.

Second, to allow DNS servers to tell which reply goes with which request, all requests carry a sequence number. To spoof Alice's ISP, Trudy has to know its current sequence number. The easiest way to learn the current sequence number is for Trudy to register a domain herself, say, trudy-the-intruder.com. Let us assume its IP address is also 42.9.9.9. She also creates a DNS server for her newly-hatched domain, dns.trudy-the-intruder.com. It, too, uses Trudy's 42.9.9.9 IP address, since Trudy has only one computer. Now she has to make Alice's ISP aware of her DNS server. That is easy to do. All she has to do is ask Alice's ISP for foobar.trudy-the-intruder.com, which will cause Alice's ISP to find out who serves Trudy's new domain by asking the top-level com server.

With dns.trudy-the-intruder.com safely in the cache at Alice's ISP, the real attack can start. Trudy now queries Alice's ISP for www.trudy-the-intruder.com. The ISP naturally sends Trudy's DNS server a query asking for it. This query bears the sequence number that Trudy is looking for. Quick like a bunny, Trudy asks Alice's ISP to look up Bob. She immediately answers her own question by sending the ISP a forged reply, allegedly from the top-level com server saying: "bob.com is 42.9.9.9". This forged

reply carries a sequence number one higher than the one she just received. While she is at it, she can also send a second forgery with a sequence number two higher, and maybe a dozen more with increasing sequence numbers. One of them is bound to match. The rest will just be thrown out. When Alice's forged reply arrives, it is cached; when the real reply comes in later, it is rejected since no query is then outstanding.

Now when Alice looks up bob.com, she is told to use 42.9.9.9, Trudy's address. Trudy has mounted a successful man-in-the-middle attack from the comfort of her own living room. The various steps to this attack are illustrated in Figure 5.47. To make matters worse, this is not the only way to spoof DNS. There are many other ways as well.

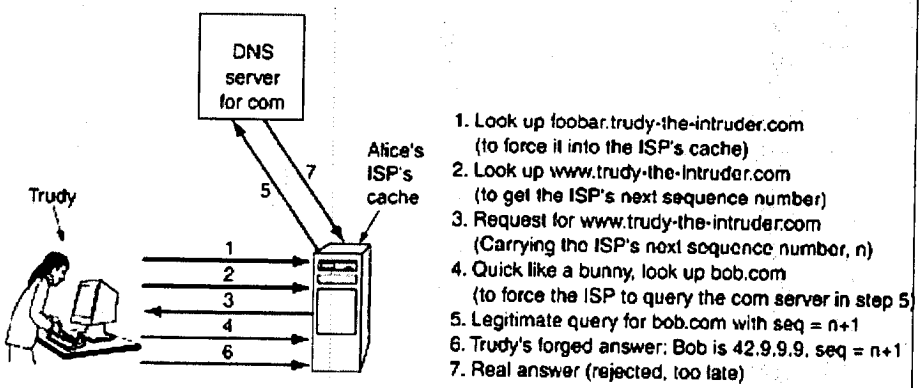


Figure 5.47. How Trudy spoofs Alice's ISP.

Secure DNS

This one specific attack can be foiled by having DNS servers use random IDs in their queries rather than just counting, but it seems that every time one hole is plugged, another one turns up. The real problem is that DNS was designed at a time when the Internet was a research facility for a few hundred universities and neither Alice, nor Bob, nor Trudy was invited to the party. Security was not an issue then; making the Internet work at all was the issue. The environment has changed radically over the years, so in 1994 IETF set up a working group to make DNS fundamentally secure. This project is known as DNSsec (DNS security); its output is presented in RFC 2535. Unfortunately, DNSsec has not been fully deployed yet, so numerous DNS servers are still vulnerable to spoofing attacks.

DNSsec is conceptually extremely simple. It is based on public-key cryptography. Every DNS zone has a public/private key pair. All information

sent by a DNS server is signed with the originating zone's private key, so the receiver can verify its authenticity.

DNSsec offers three fundamental services:

1. Proof of where the data originated.
2. Public key distribution.
3. Transaction and request authentication.

The main service is the first one, which verifies that the data being returned has been approved by the zone's owner. The second one is useful for storing and retrieving public keys securely. The third one is needed to guard against playback and spoofing attacks. Note that secrecy is not an offered service since all the information in DNS is considered public. Since phasing in DNSsec is expected to take several years, the ability for security-aware servers to interwork with security-ignorant servers is essential, which implies that the protocol cannot be changed. Let us now look at some of the details.

DNS records are grouped into sets called RRSets (Resource Record Sets), with all the records having the same name, class and type being lumped together in a set. An RRSet may contain multiple A records, for example, if a DNS name resolves to a primary IP address and a secondary IP address. The RRSets are extended with several new record types (discussed below). Each RRSet is cryptographically hashed (e.g., using MD5 or SHA-1). The hash is signed by the zone's private key (e.g., using RSA). The unit of transmission to clients is the signed RRSet. Upon receipt of a signed RRSet, the client can verify whether it was signed by the private key of the originating zone. If the signature agrees, the data are accepted. Since each RRSet contains its own signature, RRSets can be cached anywhere, even at untrustworthy servers, without endangering the security.

DNSsec introduces several new record types. The first of these is the KEY record. This records holds the public key of a zone, user, host, or other principal, the cryptographic algorithm used for signing, the protocol used for transmission, and a few other bits. The public key is stored naked. X.509 certificates are not used due to their bulk. The algorithm field holds a 1 for MD5/RSA signatures (the preferred choice), and other values for other combinations. The protocol field can indicate the use of IPsec or other security protocols, if any.

The second new record type is the SIG record. It holds the signed hash according to the algorithm specified in the KEY record The signature applies to all the records in the RRSet, including any KEY records present, but excluding itself. It also holds the times when the signature begins its period of validity and when it expires, as well as the signer's name and a few other items.

The DNSsec design is such that a zone's private key can be kept off-line. Once or twice a day, the contents of a zone's database can be manually transported (e.g., on CD-ROM) to a disconnected machine on which the private key is located. All the RRSets can be signed there and the SIG records thus produced can be conveyed back to the zone's primary server on CD-ROM. In this way, the private key can be stored on a CD-ROM locked in a safe except when it is inserted into the disconnected machine for signing the day's new RRSets. After signing is completed, all copies of the key are erased from memory and the disk and the CD-ROM are returned to the safe. This procedure reduces electronic security to physical security, something people understand how to deal with.

This method of presigning RRSets greatly speeds up the process of answering queries since no cryptography has to be done on the fly. The trade-off is that a large amount of disk space is needed to store all the keys and signatures in the DNS databases. Some records will increase tenfold in size due to the signature.

When a client process gets a signed RRSet, it must apply the originating zone's public key to decrypt the hash, compute the hash itself, and compare the two values. If they agree, the data are considered valid. However, this procedure begs the question of how the client gets the zone's public key. One way is to acquire it from a trusted server, using a secure connection (e.g., using IPsec).

However, in practice, it is expected that clients will be preconfigured with the public keys of all the top-level domains. If Alice now wants to visit Bob's Web site, she can ask DNS for the RRSet of bob.com, which will contain his IP address and a KEY record containing Bob's public key. This RRSet will be signed by the top-level com domain, so Alice can easily verify its validity. An example of what this RRSet might contain is shown in Figure 5.48.

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

Figure 5.48. An example RRSet for bob.com. The KEY record is Bob's public key. The SIG record is the top-level com server's signed hash of the A and KEY records to verify their authenticity.

Now armed with a verified copy of Bob's public key, Alice can ask Bob's DNS server (run by Bob) for the IP address of www.bob.com. This RRSet will be signed by Bob's private key, so Alice can verify the signature on the RRSet Bob returns. If Trudy somehow manages to inject a false RRSet into any of the caches, Alice can easily detect its lack of authenticity because the SIG record contained in it will be incorrect.

However, DNSsec also provides a cryptographic mechanism to bind a response to a specific query, to prevent the kind of spoof Trudy managed to pull off in Figure 5.47. This (optional) antispoofing measure adds to the response a hash of the query message signed with the respondent's private key. Since Trudy does not know the private key of the top-level com server, she cannot forge a response to a query Alice's ISP sent there. She can certainly get her response back first, but it will be rejected due to its invalid signature over the hashed query.

DNSsec also supports a few other record types. For example, the CERT record can be used for storing (e.g., X.509) certificates. This record has been provided because some people want to turn DNS into a PKI. Whether this actually happens remains to be seen. We will stop our discussion of DNSsec here. For more details, please consult RFC 2535.

Self-Certifying Names

Secure DNS is not the only possibility for securing names. A completely different approach is used in the Secure File System. In this project, the authors designed a secure, scalable, worldwide file system, without modifying (standard) DNS and without using certificates or assuming the existence of a PKI. In this section we will show how their ideas could be applied to the Web. Accordingly, in the description below, we will use Web terminology rather than the file system terminology used in the paper. But to avoid any possible confusion, while this scheme could be applied to the

Web to achieve high security, it is not currently in use and would require substantial software changes to introduce it.

We start out by assuming that each Web server has a public/private key pair. The essence of the idea is that each URL contains a cryptographic hash of the server's name and public key as part of the URL. For example, in Figure 5.49 we see the URL for Bob's photo. It starts out with the usual http scheme followed by the DNS name of the server (www.bob.com). Then comes a colon and 32-character hash. At the end is the name of the file, again as usual. Except for the hash, this is a standard URL. With the hash, it is a self-certifying URL.

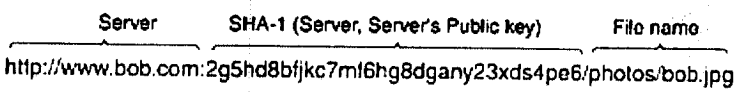


Figure 5.49. A self-certifying URL containing a hash of server's name and public key.

The obvious question is: What is the hash for? The hash is computed by concatenating the DNS name of the server with the server's public key and running the result through the SHA-1 function to get a 160-bit hash. In this scheme, the hash is represented as a sequence of 32 digits and lower-case letters, with the exception of the letters "l" and "o" and the digits "1" and "0", to avoid confusion. This leaves 32 digits and letters over. With 32 characters available, each one can encode a 5-bit string. A string of 32 characters can hold the 160-bit SHA-1 hash. Actually, it is not necessary to use a hash; the key itself could be used. The advantage of the hash is to reduce the length of the name.

In the simplest (but least convenient) way to see Bob's photo, Alice just types the string of Figure 5.49 to her browser. The browser sends a message to Bob's Web site asking him for his public key. When Bob's public key arrives, the browser concatenates the server name and public key and runs the hash algorithm. If the result agrees with the 32-character hash in the secure URL, the browser is sure it has Bob's public key. After all, due to the properties of SHA-1, even if Trudy intercepts the request and forges the reply, she has no way to find a public key that gives the expected hash. Any interference from her will thus be detected. Bob's public key can be cached for future use.

Now Alice has to verify that Bob has the corresponding private key. She constructs a message containing a proposed AES session key, a nonce, and a timestamp. She then encrypts the message with Bob's public key and sends it to him. Since only Bob has the corresponding private key, only Bob is able to decrypt the message and send back the nonce encrypted with the AES key. Upon receiving the correct AES-encrypted nonce, Alice knows she is talking to Bob. Also, Alice and Bob now have an AES session key for subsequent GET requests and replies.

Once Alice has Bob's photo (or any Web page), she can bookmark it, so she does not have to type in the full URL again. Furthermore, the URLs embedded in Web pages can also be self certifying, so they can be used by just clicking on them, but with the additional security of knowing that the page returned is the correct one. Other ways to avoid the initial typing of the self-certifying URLs are to get them over a secure connection to a trusted server or have them present in X.509 certificates signed by CAs.

Another way to get self-certifying URLs would be to connect to a trusted search engine by typing in its self-certifying URL (the first time) and going through the same protocol as described above, leading to a secure, authenticated connection to the trusted search engine. The search engine could then be queried, with the results appearing on a signed page full of self-certifying URLs that could be clicked on without having to type in long strings.

Let us now see how well this approach stands up to Trudy's DNS spoofing. If Trudy manages to poison the cache of Alice's ISP, Alice's request may be falsely delivered to Trudy rather than to Bob. But the protocol now requires the recipient of an initial message (i.e., Trudy) to return a public key that produces the correct hash. If Trudy returns her own public key, Alice will detect it immediately because the SHA-1 hash will not match the self-certifying URL. If Trudy returns Bob's public key, Alice will not detect the attack, but Alice will encrypt her next message, using Bob's key. Trudy will get the message, but she will have no way to decrypt it to extract the AES key and nonce. Either way, all spoofing DNS can do is provide a denial-of-service attack.

SSL—The Secure Sockets Layer

Secure naming is a good start, but there is much more to Web security. The next step is secure connections. We will now look at how secure connections can be achieved.

When the Web burst into public view, it was initially used for just distributing static pages. However, before long, some companies got the idea of using it for financial transactions, such as purchasing merchandise by credit card, on-line banking, and electronic stock trading. These applications created a demand for secure connections. In 1995, Netscape Communications Corp, the then-dominant browser vendor, responded by introducing a security package called SSL (Secure Sockets Layer) to meet this demand. This software and its protocol is now widely used, also by Internet Explorer, so it is worth examining in some detail.

SSL builds a secure connection between two sockets, including

1. Parameter negotiation between client and server.
2. Mutual authentication of client and server.
3. Secret communication.
4. Data integrity protection.

We have seen these items before, so there is no need to elaborate on them.

The positioning of SSL in the usual protocol stack is illustrated in Figure 5.50. Effectively, it is a new layer interposed between the application layer and the transport layer, accepting requests from the browser and sending them down to TCP for transmission to the server. Once the secure connection has been established, SSL's main job is handling compression and encryption. When HTTP is used over SSL, it is called HTTPS (Secure HTTP), even though it is the standard HTTP protocol. Sometimes it is available at a new port (443) instead of the standard port (80), though. As an aside, SSL is not restricted to being used only with Web browsers, but that is its most common application.

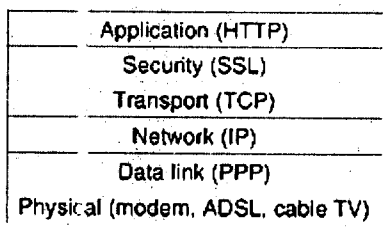


Figure 5.50. Layers (and protocols) for a home user browsing with SSL.

The SSL protocol has gone through several versions. Below we will discuss only version 3, which is the most widely used version. SSL supports a variety of different algorithms and options. These options include the presence or absence of compression, the cryptographic algorithms to be used, and some matters relating to export restrictions on cryptography. The last is mainly intended to make sure that serious cryptography is used only when both ends of the connection are in the United States. In other cases, keys are limited to 40 bits, which cryptographers regard as something of a joke. Netscape was forced to put in this restriction in order to get an export license from the U.S. Government.

SSL consists of two subprotocols, one for establishing a secure connection and one for using it. Let us start out by seeing how secure connections are established. The connection establishment subprotocol is shown in Figure 5.5. It starts out with message 1 when Alice sends a request to Bob to establish a connection. The request specifies the SSL version Alice has and her preferences with respect to compression and cryptographic algorithms. It also contains a nonce, R_A, to be used later.

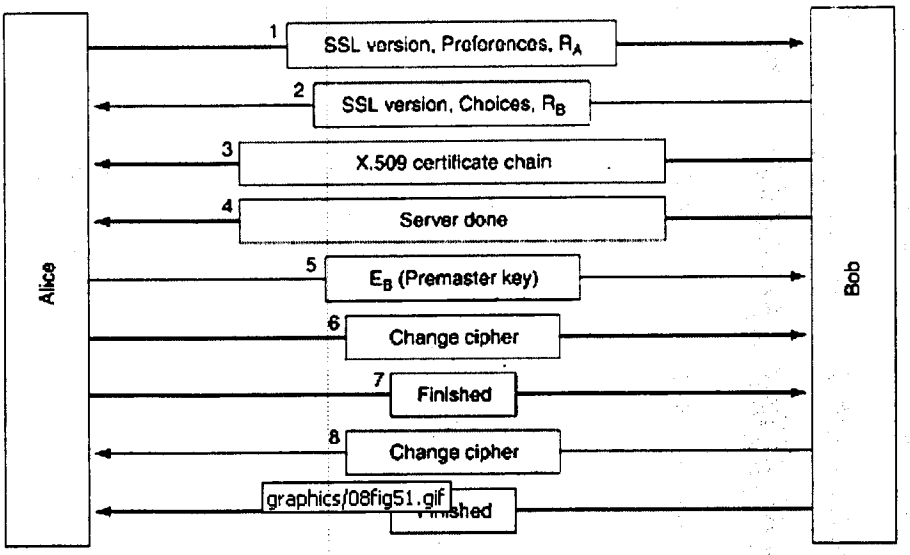


Figure 5.51. A simplified version of the SSL connection establishment subprotocol.

Now it is Bob's turn. In message 2, Bob makes a choice among the various algorithms that Alice can support and sends his own nonce, R_B. Then in message 3, he sends a certificate containing his public key. If this certificate is not signed by some well-known authority, he also sends a chain of certificates that can be followed back to one. All browsers, including Alice's, come preloaded with about 100 public keys, so if Bob can establish a chain anchored at one of these, Alice will be able to verify Bob's public key. At this point Bob may send some other messages (such as a request for Alice's public-key certificate). When Bob is done, he sends message 4 to tell Alice it is her turn.

Alice responds by choosing a random 384-bit premaster key and sending it to Bob encrypted with his public key (message 5). The actual session key used for encrypting data is derived from the premaster key combined with both nonces in a complex way. After message 5 has been received, both Alice and Bob are able to compute the session key. For this reason, Alice tells Bob to switch to the new cipher (message 6) and also that she is finished with the establishment subprotocol (message 7). Bob then acknowledges her (messages 8 and 9).

However, although Alice knows who Bob is, Bob does not know who Alice is (unless Alice has a public key and a corresponding certificate for it, an unlikely situation for an individual). Therefore, Bob's first message may well

be a request for Alice to log in using a previously established login name and password. The login protocol, however, is outside the scope of SSL. Once it has been accomplished, by whatever means, data transport can begin.

As mentioned above, SSL supports multiple cryptographic algorithms. The strongest one uses triple DES with three separate keys for encryption and SHA-1 for message integrity. This combination is relatively slow, so it is mostly used for banking and other applications in which the highest security is required. For ordinary e-commerce applications, RC4 is used with a 128-bit key for encryption and MD5 is used for message authentication. RC4 takes the 128-bit key as a seed and expands it to a much larger number for internal use. Then it uses this internal number to generate a keystream. The keystream is XORed with the plaintext to provide a classical stream cipher, as we saw in Figure 5.14. The export versions also use RC4 with 128-bit keys, but 88 of the bits are made public to make the cipher easy to break.

For actual transport, a second subprotocol is used, as shown in Figure 5.52. Messages from the browser are first broken into units of up to 16 KB. If compression is enabled, each unit is then separately compressed. After that, a secret key derived from the two nonces and premaster key is concatenated with the compressed text and the result hashed with the agreed-on hashing algorithm (usually MD5). This hash is appended to each fragment as the MAC. The compressed fragment plus MAC is then encrypted with the agreed-on symmetric encryption algorithm (usually by XORing it with the RC4 keystream). Finally, a fragment header is attached and the fragment is transmitted over the TCP connection.

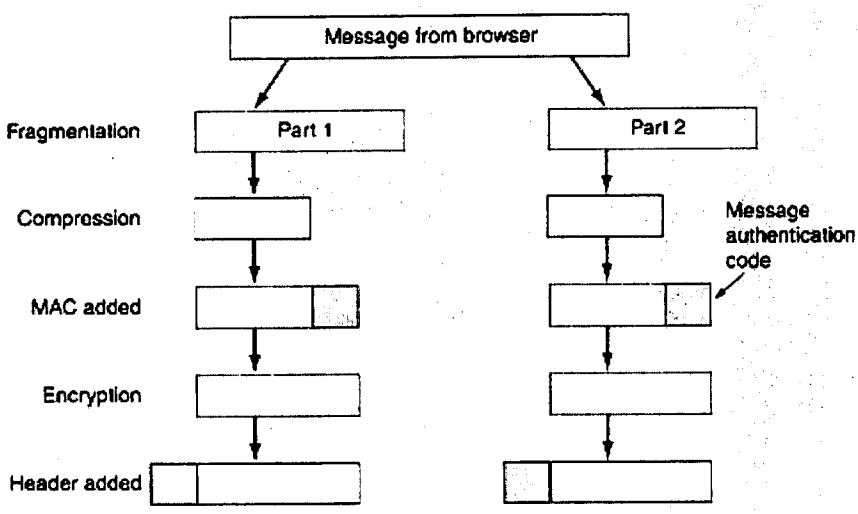


Figure 5.52. Data transmission using SSL.

A word of caution is in order, however. Since it has been shown that RC4 has some weak keys that can be easily cryptanalyzed, the security of SSL using RC4 is on shaky ground. Browsers that allow the user to choose the cipher suite should be configured to use triple DES with 168-bit keys and SHA-1 all the time, even though this combination is slower than RC4 and MD5.

Another problem with SSL is that the principals may not have certificates and even if they do, they do not always verify that the keys being used match them.

In 1996, Netscape Communications Corp. turned SSL over to IETF for standardization. The result was TLS (Transport Layer Security). It is described in RFC 2246.

The changes made to SSL were relatively small, but just enough that SSL version 3 and TLS cannot interoperate. For example, the way the session key is derived from the premaster key and nonces was changed to make the key stronger (i.e., harder to cryptanalyze). The TLS version is also known as SSL version 3.1. The first implementations appeared in 1999, but it is not clear yet whether TLS will replace SSL in practice, even though it is slightly stronger. The problem with weak RC4 keys remains, however.

Mobile Code Security

Naming and connections are two areas of concern related to Web security. But there are more. In the early days, when Web pages were just static HTML files, they did not contain executable code. Now they often contain small programs, including Java applets, ActiveX controls, and JavaScripts. Downloading and executing such mobile code is obviously a massive security risk, so various methods have been devised to minimize it. We will now take a quick peek at some of the issues raised by mobile code and some approaches to dealing with it.

Java Applet Security

Java applets are small Java programs compiled to a stack-oriented machine language called JVM (Java Virtual Machine). They can be placed on a Web page for downloading along with the page. After the page is loaded, the applets are inserted into a JVM interpreter inside the browser, as illustrated in Figure 5.53.

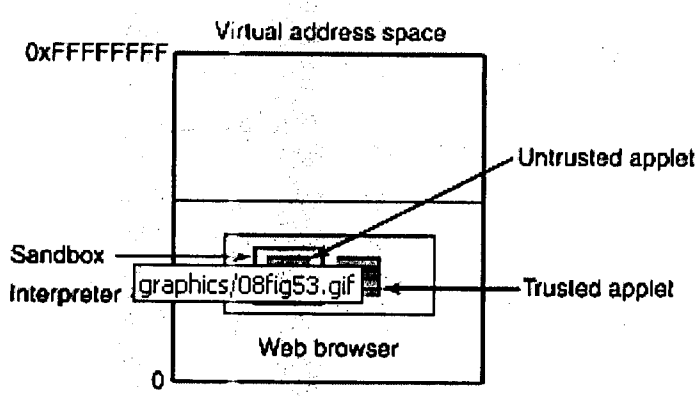


Figure 5.53. Applets can be interpreted by a Web browser.

The advantage of running interpreted code over compiled code is that every instruction is examined by the interpreter before being executed. This gives the interpreter the opportunity to check whether the instruction's address is valid. In addition, system calls are also caught and interpreted. How these calls are handled is a matter of the security policy. For example, if an applet is trusted (e.g., it came from the local disk), its system calls could be carried out without question. However, if an applet is not trusted (e.g., it came in over the Internet), it could be encapsulated in what is called

a sandbox to restrict its behavior and trap its attempts to use system resources.

When an applet tries to use a system resource, its call is passed to a security monitor for approval. The monitor examines the call in light of the local security policy and then makes a decision to allow or reject it. In this way, it is possible to give applets access to some resources but not all. Unfortunately, the reality is that the security model works badly and that bugs in it crop up all the time.

ActiveX

ActiveX controls are Pentium binary programs that can be embedded in Web pages. When one of them is encountered, a check is made to see if it should be executed, and it if passes the test, it is executed. It is not interpreted or sandboxed in any way, so it has as much power as any other user program and can potentially do great harm. Thus, all the security is in the decision whether to run the ActiveX control.

The method that Microsoft chose for making this decision is based on the idea of code signing. Each ActiveX control is accompanied by a digital signature a hash of the code that is signed by its creator using public key cryptography. When an ActiveX control shows up, the browser first verifies the signature to make sure it has not been tampered with in transit. If the signature is correct, the browser then checks its internal tables to see if the program's creator is trusted or there is a chain of trust back to a trusted creator. If the creator is trusted, the program is executed; otherwise, it is not. The Microsoft system for verifying ActiveX controls is called Authenticode.

It is useful to contrast the Java and ActiveX approaches. With the Java approach, no attempt is made to determine who wrote the applet. Instead, a run-time interpreter makes sure it does not do things the machine owner has said applets may not do. In contrast, with code signing, there is no attempt to monitor the mobile code's run-time behavior. If it came from a trusted source and has not been modified in transit, it just runs. No attempt is made to see whether the code is malicious or not. If the original programmer intended the code to format the hard disk and then erase the flash ROM so the computer can never again be booted, and if the programmer has been certified as trusted, the code will be run and destroy the computer (unless ActiveX controls have been disabled in the browser).

Many people feel that trusting an unknown software company is scary. To demonstrate the problem, a programmer in Seattle formed a software company and got it certified as trustworthy, which is easy to do. He then wrote an ActiveX control that did a clean shutdown of the machine and distributed his ActiveX control widely. It shut down many machines, but they could just be rebooted, so no harm was done. He was just trying to expose the problem to the world. The official response was to revoke the certificate for this specific ActiveX control, which ended a short episode of acute embarrassment, but the underlying problem is still there for an evil programmer to exploit. Since there is no way to police thousands of software companies that might write mobile code, the technique of code signing is a disaster waiting to happen.

JavaScript

JavaScript does not have any formal security model, but it does have a long history of leaky implementations. Each vendor handles security in a different way. For example, Netscape Navigator version 2 used something akin to the Java model, but by version 4 that had been abandoned for a code signing model.

The fundamental problem is that letting foreign code run on your machine is asking for trouble. From a security standpoint, it is like inviting a burglar into your house and then trying to watch him carefully so he cannot escape from the kitchen into the living room. If something unexpected happens and you are distracted for a moment, bad things can happen. The tension here is that mobile code allows flashy graphics and fast interaction, and many Web site designers think that this is much more important than security, especially when it is somebody else's machine at risk.

Viruses

Viruses are another form of mobile code. Only unlike the examples above, viruses are not invited in at all. The difference between a virus and ordinary mobile code is that viruses are written to reproduce themselves. When a virus arrives, either via a Web page, an e-mail attachment, or some other way, it usually starts out by infecting executable programs on the disk. When one of these programs is run, control is transferred to the virus, which usually tries to spread itself to other machines, for example, by e-mailing copies of itself to everyone in the victim's e-mail address book. Some viruses infect the boot sector of the hard disk, so when the machine

is booted, the virus gets to run. Viruses have become a huge problem on the Internet and have caused billions of dollars worth of damage. There is no obvious solution. Perhaps a whole new generation of operating systems based on secure microkernels and tight compartmentalization of users, processes, and resources might help.

5.10 Social Issues

The Internet and its security technology is an area where social issues, public policy, and technology meet head on, often with huge consequences. Below we will just briefly examine three areas: privacy, freedom of speech, and copyright. Needless to say, we can only scratch the surface here. The Internet is also full of material. Just type words such as "privacy," "censorship," and "copyright" into any search engine. Also, see this book's Web site for some links.

Privacy

Do people have a right to privacy? Good question. The Fourth Amendment to the U.S. Constitution prohibits the government from searching people's houses, papers, and effects without good reason, and goes on to restrict the circumstances under which search warrants shall be issued. Thus, privacy has been on the public agenda for over 200 years, at least in the U.S.

What has changed in the past decade is both the ease with which governments can spy on their citizens and the ease with which the citizens can prevent such spying. In the 18th century, for the government to search a citizen's papers, it had to send out a policeman on a horse to go to the citizen's farm demanding to see certain documents. It was a cumbersome procedure. Nowadays, telephone companies and Internet providers readily provide wiretaps when presented with search warrants. It makes life much easier for the policeman and there is no danger of falling off the horse.

Cryptography changes all that. Anybody who goes to the trouble of downloading and installing PGP and who uses a well-guarded alien-strength key can be fairly sure that nobody in the known universe can read his e-mail, search warrant or no search warrant. Governments well understand this and do not like it. Real privacy means it is much harder for them to spy on criminals of all stripes, but it is also much harder to spy on journalists and political opponents. Consequently, some governments

restrict or forbid the use or export of cryptography. In France, for example, prior to 1999, all cryptography was banned unless the government was given the keys.

France was not alone. In April 1993, the U.S. Government announced its intention to make a hardware cryptoprocessor, the clipper chip, the standard for all networked communication. In this way, it was said, citizens' privacy would be guaranteed. It also mentioned that the chip provided the government with the ability to decrypt all traffic via a scheme called key escrow, which allowed the government access to all the keys. However, it promised only to snoop when it had a valid search warrant. Needless to say, a huge furor ensued, with privacy advocates denouncing the whole plan and law enforcement officials praising it. Eventually, the government backed down and dropped the idea.

Anonymous Remailers

PGP, SSL, and other technologies make it possible for two parties to establish secure, authenticated communication, free from third-party surveillance and interference. However, sometimes privacy is best served by not having authentication, in fact by making communication anonymous. The anonymity may be desired for point-to-point messages, newsgroups, or both.

Let us consider some examples. First, political dissidents living under authoritarian regimes often wish to communicate anonymously to escape being jailed or killed. Second, wrongdoing in many corporate, educational, governmental, and other organizations has often been exposed by whistleblowers, who frequently prefer to remain anonymously to avoid retribution. Third, people with unpopular social, political, or religious views may wish to communicate with each other via e-mail or newsgroups without exposing themselves. Fourth, people may wish to discuss alcoholism, mental illness, sexual harassment, child abuse, or being a member of a persecuted minority in a newsgroup without having to go public. Numerous other examples exist, of course.

Let us consider a specific example. In the 1990s, some critics of a nontraditional religious group posted their views to a USENET newsgroup via an anonymous remailer. This server allowed users to create pseudonyms and send e-mail to the server, which then re-mailed or re-posted them using the pseudonym, so no one could tell where the

message really came from. Some postings revealed what the religious group claimed were trade secrets and copyrighted documents. The religious group responded by telling local authorities that its trade secrets had been disclosed and its copyright infringed, both of which were crimes where the server was located. A court case followed and the server operator was compelled to turn over the mapping information which revealed the true identities of the persons who had made the postings. (Incidentally, this was not the first time that a religion was unhappy when someone leaked its secrets: William Tyndale was burned at the stake in 1536 for translating the Bible into English).

A substantial segment of the Internet community was outraged by this breach of confidentiality. The conclusion that everyone drew is that an anonymous remailer that stores a mapping between real e-mail addresses and pseudonyms (called a type 1 remailer) is not worth much. This case stimulated various people into designing anonymous remailers that could withstand subpoena attacks.

These new remailers, often called cypherpunk remailers, work as follows. The user produces an e-mail message, complete with RFC 822 headers (except From:, of course), encrypts it with the remailer's public key, and sends it to the remailer. There the outer RFC 822 headers are stripped off, the content is decrypted and the message is remailed. The remailer has no accounts and maintains no logs, so even if the server is later confiscated, it retains no trace of messages that have passed through it.

Many users who wish anonymity chain their requests through multiple anonymous remailers, as shown in Figure 5.54. Here, Alice wants to send Bob a really, really, really anonymous Valentine's Day card, so she uses three remailers. She composes the message, M, and puts a header on it containing Bob's e-mail address. Then she encrypts the whole thing with remailer 3's public key, E_3. (indicated by horizontal hatching). To this she prepends a header with remailer 3's e-mail address in plaintext. This is the message shown between remailers 2 and 3 in the figure.

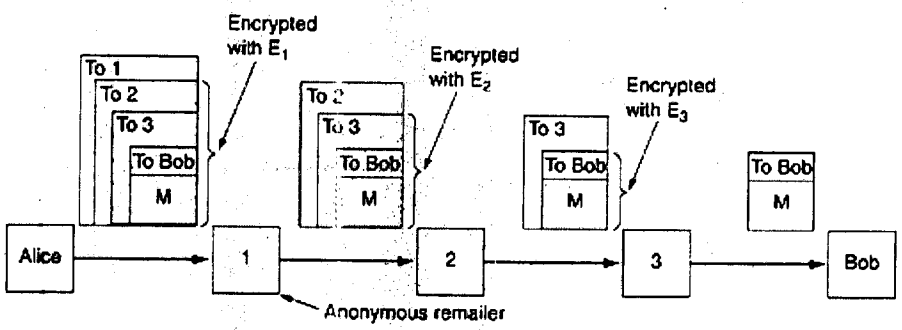


Figure 5.54. How Alice uses 3 remailers to send Bob a message.

Then she encrypts this message with remailer 2's public key, E_2 (indicated by vertical hatching) and prepends a plaintext header containing remailer 2's e-mail address. This message is shown between 1 and 2 in Figure 5.54. Finally, she encrypts the entire message with remailer 1's public key, E_1, and prepends a plaintext header with remailer 1's e-mail address. This is the message shown to the right of Alice in the figure and this is the message she actually transmits.

When the message hits remailer 1, the outer header is stripped off. The body is decrypted and then e-mailed to remailer 2. Similar steps occur at the other two remailers.

Although it is extremely difficult for anyone to trace the final message back to Alice, many remailers take additional safety precautions. For example, they may hold messages for a random time, add or remove junk at the end of a message, and reorder messages, all to make it harder for anyone to tell which message output by a remailer corresponds to which input, in order to thwart traffic analysis. For a description of a system that represents the state of the art in anonymous e-mail.

Anonymity is not restricted to e-mail. Services also exist that allow anonymous Web surfing. The user configures his browser to use the anonymizer as a proxy. Henceforth, all HTTP requests go to the anonymizer, which requests the page and sends it back. The Web site sees the anonymizer as the source of the request, not the user. As long as the anonymizer refrains from keeping a log, after the fact no one can determine who requested which page.

Freedom of Speech

Privacy relates to individuals wanting to restrict what other people can see about them. A second key social issue is freedom of speech, and its opposite, censorship, which is about governments wanting to restrict what individuals can read and publish. With the Web containing millions and millions of pages, it has become a censor's paradise. Depending on the nature and ideology of the regime, banned material may include Web sites containing any of the following:

1. Material inappropriate for children or teenagers.
2. Hate aimed at various ethnic, religious, sexual or other groups.
3. Information about democracy and democratic values.
4. Accounts of historical events contradicting the government's version.
5. Manuals for picking locks, building weapons, encrypting messages, etc.

The usual response is to ban the bad sites.

Sometimes the results are unexpected. For example, some public libraries have installed Web filters on their computers to make them child friendly by blocking pornography sites. The filters veto sites on their blacklists but also check pages for dirty words before displaying them. In one case in Loudoun County, Virginia, the filter blocked a patron's search for information on breast cancer because the filter saw the word "breast." The library patron sued Loudoun county. However, in Livermore, California, a parent sued the public library for not installing a filter after her 12-year-old son was caught viewing pornography there. What's a library to do?

It has escaped many people that the World Wide Web is a Worldwide Web. It covers the whole world. Not all countries agree on what should be allowed on the Web. For example, in November 2000, a French court ordered Yahoo, a California Corporation, to block French users from viewing auctions of Nazi memorabilia on Yahoo's Web site because owning such material violates French law. Yahoo appealed to a U.S. court, which sided with it, but the issue of whose laws apply where is far from settled.

Just imagine. What would happen if some court in Utah instructed France to block Web sites dealing with wine because they do not comply with Utah's much stricter laws about alcohol? Suppose that China demanded that all Web sites dealing with democracy be banned as not in the interest of the State. Do Iranian laws on religion apply to more liberal Sweden? Can Saudi Arabia block Web sites dealing with women's rights? The whole issue is a veritable Pandora's box.

A relevant comment from John Gilmore is: "The net interprets censorship as damage and routes around it." For a concrete implementation, consider the eternity service (Anderson, 1996). Its goal is make sure published information cannot be depublished or rewritten, as was common in the Soviet Union during Josef Stalin's reign. To use the eternity service, the user specifies how long the material is to be preserved, pays a fee proportional to its duration and size, and uploads it. Thereafter, no one can remove or edit it, not even the uploader.

How could such a service be implemented? The simplest model is to use a peer-to-peer system in which stored documents would be placed on dozens of participating servers, each of which gets a fraction of the fee, and thus an incentive to join the system. The servers should be spread over many legal jurisdictions for maximum resilience. Lists of 10 randomly-selected servers would be stored securely in multiple places, so that if some were compromised, others would still exist. An authority bent on destroying the document could never be sure it had found all copies. The system could also be made self-repairing in the sense that if it became known that some copies had been destroyed, the remaining sites would attempt to find new repositories to replace them.

The eternity service was the first proposal for a censorship-resistant system. Since then, others have been proposed and, in some cases, implemented. Various new features have been added, such as encryption, anonymity, and fault tolerance. Often the files to be stored are broken up into multiple fragments, with each fragment stored on many servers.

Increasingly, many countries are now trying to regulate the export of intangibles, which often include Web sites, software, scientific papers, e-mail, telephone helpdesks, and more. Even the U.K., which has a centuries-long tradition of freedom of speech, is now seriously considering highly restrictive laws, which would, for example, define technical discussions between a British professor and his foreign student at the

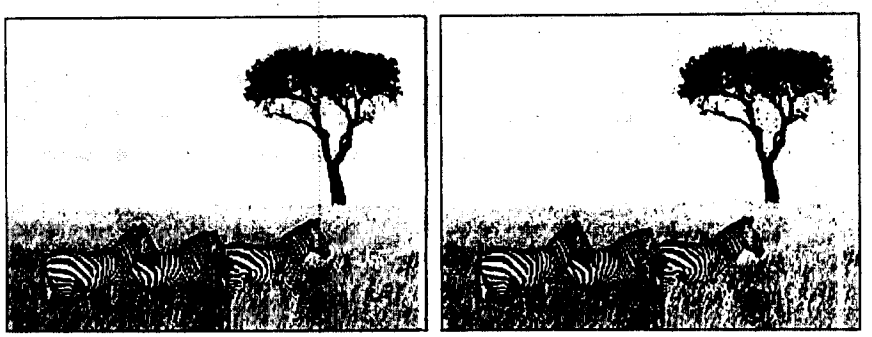
University of Cambridge as regulated export needing a government license. Needless to say, such policies are controversial.

Steganography

In countries where censorship abounds, dissidents often try to use technology to evade it. Cryptography allows secret messages to be sent (although possibly not lawfully), but if the government thinks that Alice is a Bad Person, the mere fact that she is communicating with Bob may get him put in this category, too, as repressive governments understand the concept of transitive closure, even if they are short on mathematicians. Anonymous remailers can help, but if they are banned domestically and messages to foreign ones require a government export license, they cannot help much. But the Web can.

People who want to communicate secretly often try to hide the fact that any communication at all is taking place. The science of hiding messages is called steganography, from the Greek words for "covered writing." In fact, the ancient Greeks used it themselves. Herodotus wrote of a general who shaved the head of a messenger, tattooed a message to his scalp, and let the hair grow back before sending him off. Modern techniques are conceptually the same, only they have a higher bandwidth and lower latency.

As a case in point, consider Figure 5.55(a). This photograph, taken by the author in Kenya, contains three zebras contemplating an acacia tree. Figure 5.55(b) appears to be the same three zebras and acacia tree, but it has an extra added attraction. It contains the complete, unabridged text of five of Shakespeare's plays embedded in it: Hamlet, King Lear, Macbeth, The Merchant of Venice, and Julius Caesar. Together, these plays total over 700 KB of text.



(a) (b)

Figure 5.55. (a) Three zebras and a tree. (b) Three zebras, a tree, and the complete text of five plays by William Shakespeare.

How does this steganographic channel work? The original color image is 1024 x 768 pixels. Each pixel consists of three 8-bit numbers, one each for the red, green, and blue intensity of that pixel. The pixel's color is formed by the linear superposition of the three colors. The steganographic encoding method uses the low-order bit of each RGB color value as a covert channel. Thus, each pixel has room for 3 bits of secret information, one in the red value, one in the green value, and one in the blue value. With an image of this size, up to 1024 x 768 x 3 bits or 294,912 bytes of secret information can be stored in it.

The full text of the five plays and a short notice add up to 734,891 bytes. This text was first compressed to about 274 KB using a standard compression algorithm. The compressed output was then encrypted using IDEA and inserted into the low-order bits of each color value. As can be seen (or actually, cannot be seen), the existence of the information is completely invisible. It is equally invisible in the large, full-color version of the photo. The eye cannot easily distinguish 21-bit color from 24-bit color.

Viewing the two images in black and white with low resolution does not do justice to how powerful the technique is. To get a better feel for how steganography works, the author has prepared a demonstration, including the full-color high-resolution image of Fig. 8-55(b) with the five plays embedded in it. The demonstration, including tools for inserting and extracting text into images, can be found at the book's Web site.

To use steganography for undetected communication, dissidents could create a Web site bursting with politically-correct pictures, such as photographs of the Great Leader, local sports, movie, and television stars, etc. Of course, the pictures would be riddled with steganographic messages. If the messages were first compressed and then encrypted, even someone who suspected their presence would have immense difficulty in distinguishing the messages from white noise. Of course, the images should be fresh scans; copying a picture from the Internet and changing some of the bits is a dead giveaway.

Images are by no means the only carrier for steganographic messages. Audio files also work fine. Video files have a huge steganographic bandwidth. Even the layout and ordering of tags in an HTML file can carry information.

Although we have examined steganography in the context of free speech, it has numerous other uses. One common use is for the owners of images to encode secret messages in them stating their ownership rights. If such an image is stolen and placed on a Web site, the lawful owner can reveal the steganographic message in court to prove whose image it is. This technique is called watermarking.

Copyright

Privacy and censorship are just two areas where technology meets public policy. A third one is copyright. Copyright is the granting to the creators of IP (Intellectual Property), including writers, artists, composers, musicians, photographers, cinematographers, choreographers, and others, the exclusive right to exploit their IP for some period of time, typically the life of the author plus 50 years or 75 years in the case of corporate ownership. After the copyright of a work expires, it passes into the public domain and anyone can use or sell it as they wish.

Copyright came to the forefront when Napster, a music-swapping service, had 50 million members. Although Napster did not actually copy any music, the courts held that its holding a central database of who had which song was contributory infringement, that is, they helped other people infringe. While nobody seriously claims copyright is a bad idea (although many claim that the term is far too long, favoring big corporations over the public), the next generation of music sharing is already raising major ethical issues.

For example, consider a peer-to-peer network in which people share legal files (public domain music, home videos, religious tracts that are not trade secrets, etc.) and perhaps a few that are copyrighted. Assume that everyone is on-line all the time via ADSL or cable. Each machine has an index of what is on the hard disk, plus a list of other members. Someone looking for a specific item can pick a random member and see if he has it. If not, he can check out all the members in that person's list, and all the members in their lists, and so on. Computers are very good at this kind of work. Having found the item, the requester just copies it.

If the work is copyrighted, chances are the requester is infringing (although for international transfers, the question of whose law applies is unclear). But what about the supplier? Is it a crime to keep music you have paid for and legally downloaded on your hard disk where others might find it? If you have an unlocked cabin in the country and a IP thief sneaks in carrying a

notebook computer and scanner, copies a copyrighted book, and sneaks out, are you guilty of the crime of failing to protect someone else's copyright?

But there is more trouble brewing on the copyright front. There is a huge battle going on now between Hollywood and the computer industry. The former wants stringent protection of all intellectual property and the latter does not want to be Hollywood's policeman. In October 1998, Congress passed the DMCA (Digital Millennium Copyright Act) which makes it a crime to circumvent any protection mechanism present in a copyrighted work or to tell others how to circumvent it. Similar legislation is being set in place in the European Union. While virtually no one thinks that pirates in the Far East should be allowed to duplicate copyrighted works, many people think that the DMCA completely shifts the balance between the copyright owner's interest and the public interest.

A case in point. In September 2000, a music industry consortium charged with building an unbreakable system for selling music on-line sponsored a contest inviting people to try to break the system (which is precisely the right thing to do with any new security system). A team of security researchers from several universities, led by Prof. Edward Felten of Princeton, took up the challenge and broke the system. They then wrote a paper about their findings and submitted it to a USENIX security conference, where it underwent peer review and was accepted. Before the paper was to be presented, Felten received a letter from the Recording Industry Association of America which threatened to sue the authors under the DMCA if they published the paper.

Their response was to file suit asking a federal court to rule on whether publishing scientific papers on security research was still legal. Fearing a definitive court ruling against them, the industry withdrew its threat and the court dismissed Felten's suit. No doubt the industry was motivated by the weakness of its case: they had invited people to try to break their system and then threatened to sue some of them for accepting their challenge. With the threat withdrawn, the paper was published. A new confrontation is virtually certain.

A related issue is the extent of the fair use doctrine, which has been established by court rulings in various countries. This doctrine says that purchasers of a copyrighted work have certain limited rights to copy the work, including the right to quote parts of it for scientific purposes, use it as

teaching material in schools or colleges, and in some cases make backup copies for personal use in case the original medium fails. The tests for what constitutes fair use include (1) whether the use is commercial, (2) what percentage of the whole is being copied, and (3) the effect of the copying on sales of the work. Since the DMCA and similar laws within the European Union prohibit circumvention of copy protection schemes, these laws also prohibit legal fair use. In effect, the DMCA takes away historical rights from users to give content sellers more power. A major show-down is inevitable.

Another development in the works that dwarfs even the DMCA in its shifting of the balance between copyright owners and users is the TCPA (Trusted Computing Platform Alliance) led by Intel and Microsoft. The idea is to have the CPU chip and operating system carefully monitor user behavior in various ways (e.g., playing pirated music) and prohibit unwanted behavior. The system even allows content owners to remotely manipulate user PCs to change the rules when that is deemed necessary. Needless to say, the social consequences of this scheme are immense. It is nice that the industry is finally paying attention to security, but it is lamentable that it is entirely aimed at enforcing copyright law rather than dealing with viruses, crackers, intruders, and other security issues that most people are concerned about.

In short, the lawmakers and lawyers will be busy balancing the economic interests of copyright owners with the public interest for years to come. Cyberspace is no different from meatspace: it constantly pits one group against another, resulting in power struggles, litigation, and (hopefully) eventually some kind of resolution, at least until some new disruptive technology comes along.

Summary

Cryptography is a tool that can be used to keep information confidential and to ensure its integrity and authenticity. All modern cryptographic systems are based on Kerckhoff's principle of having a publicly-known algorithm and a secret key. Many cryptographic algorithms use complex transformations involving substitutions and permutations to transform the plaintext into the ciphertext. However, if quantum cryptography can be made practical, the use of one-time pads may provide truly unbreakable cryptosystems.

Cryptographic algorithms can be divided into symmetric-key algorithms and public-key algorithms. Symmetric-key algorithms mangle the bits in a series of rounds parameterized by the key to turn the plaintext into the ciphertext. Triple DES and Rijndael (AES) are the most popular symmetric-key algorithms at present. These algorithms can be used in electronic code book mode, cipher block chaining mode, stream cipher mode, counter mode, and others.

Public-key algorithms have the property that different keys are used for encryption and decryption and that the decryption key cannot be derived from the encryption key. These properties make it possible to publish the public key. The main public-key algorithm is RSA, which derives its strength from the fact that it is very difficult to factor large numbers.

Legal, commercial, and other documents need to be signed. Accordingly, various schemes have been devised for digital signatures, using both symmetric-key and public-key algorithms. Commonly, messages to be signed are hashed using algorithms such as MD5 or SHA-1, and then the hashes are signed rather than the original messages.

Public-key management can be done using certificates, which are documents that bind a principal to a public key. Certificates are signed by a trusted authority or by someone (recursively) approved by a trusted authority. The root of the chain has to be obtained in advance, but browsers generally have many root certificates built into them.

These cryptographic tools can be used to secure network traffic. IPsec operates in the network layer, encrypting packet flows from host to host. Firewalls can screen traffic going into or out of an organization, often based on the protocol and port used. Virtual private networks can simulate an old leased-line network to provide certain desirable security properties. Finally, wireless networks need good security and 802.11's WEP does not provide it, although 802.11i should improve matters considerably.

When two parties establish a session, they have to authenticate each other and if need be, establish a shared session key. Various authentication protocols exist, including some that use a trusted third party, Diffie-Hellman, Kerberos, and public-key cryptography.

E-mail security can be achieved by a combination of the techniques we have studied in this chapter. PGP, for example, compresses messages,

then encrypts them using IDEA. It sends the IDEA key encrypted with the receiver's public key. In addition, it also hashes the message and sends the signed hash to verify message integrity.

Web security is also an important topic, starting with secure naming. DNSsec provides a way to prevent DNS spoofing, as do self-certifying names. Most e-commerce Web sites use SSL to establish secure, authenticated sessions between the client and server. Various techniques are used to deal with mobile code, especially sandboxing and code signing.

The Internet raises many issues in which technology interacts strongly with public policy. Some of the areas include privacy, freedom of speech, and copyright.

Review Questions

1. In symmetric-key cryptography, how many keys are needed if A and B want to communicate with each other?

2. In symmetric-key cryptography, can A use the same key to communicate with both B and C? Explain your answer.

3. In symmetric-key cryptography, if every person in a group of 10 people needs to communicate with every other person in another group of 10 people, how many secret keys are needed?

4. In symmetric-key cryptography, if every person in a group of 10 people needs to cornmunicate with every other person in the group, how many secret keys are needed?

5. In asymmetric-key cryptography, how many keys are needed if A and B want to communicate with each other?

6. In symmetric-key cryptography, how do you think two persons can establish a secret key between themselves?

7. In asymmetric-key cryptography, how do you think two persons can establish two pairs of keys between themselves?

8. Encrypt the message "THIS IS AN EXERCISE" using a shift cipher with a key of 20. Ignore the space between words. Decrypt the message to get the original plaintext.

9. Can we use monoalphabetic substitution if our symbols are just 0 and 1? Is it a good idea?

10. Can we use polyalphabetic substitution if our symbols are just 0 and 1? Is it a good idea?

11. Suppose an organization uses Kerberos for authentication. In terms of security and service availability, what is the effect if AS or TGS goes down?

12. What is firewall? Explain its function.

Reference Books

1. Computer Networking: Schaum's outlines (TMH).

2. Kurose J F & Ross K.W: Computer Networking (Pearson)

3. Tanenbaum A S: Computer Networks (PHI) 4th Ed.

4. Data Communication & Networking – Behrouz Forouzan(TMH)

GLOSAARY

100Base-FX: A two-wire fiber implementation of Fast Ethernet.

100Base-T4: A four-wire UTP implementation of Fast Ethernet.

100Base-TX: A two-wire UTP implementation of Fast Ethernet.

10Base2: The thin coaxial cable implementation of Standard Ethernet.

10Base5: The thick coaxial cable implementation of Standard Ethernet.

10Base-F: The fiber implementation of Standard Ethernet.

10Base-T: The twisted-pair implementation of Standard Ethernet.

10Base-E: The extended implementation of Ten-Gigabit Ethernet.

10Base-L: A fiber implementation of Ten-Gigabit Ethernet using long-wave laser.

10Base-S: A fiber implementation of Ten-Gigabit Ethernet using short-wave laser.

1-persistent strategy: A CSMA persistence strategy in which a station sends frames immediately if the line is idle.

Access control: The determination of link control through a data link protocol.

Access rate: In Frame Relay, the data rate that can never be exceeded.

Access point: A central base station in a BSS.

Acknowledgment (ACK): A response sent by the receiver to indicate the successful receipt and acceptance of data.

Address Resolution Protocol (ARP): In TCPIP, a protocol for obtaining the physical address of a node when the Internet address is known.

Advanced Encryption Standard (AES): A secret-key cryptosystem adapted by NIST it replace DES.

Address-mask request and reply ICMP: Messages that find the network mask.

Advanced Research Projects Agency (ARPA): The government agency that funded ARPANET.

Advanced Research Projects Agency Network (ARPANET): The packet-switching network that was funded by ARPA.

ALOHA: The original random multiple access method in which a station can send a frame any time it has one to send.

American National Standards Institute (ANSI): A national standards organization that defines standards in the United States.

American Standard Code for Information Interchange (ASCII): A character code developed by ANSI and used extensively for data communication.

ATM switch: An ATM device providing both switching and multiplexing functions.

Attachment unit interface (AUI): A 10Base5 cable that performs the physical interface functions between the station and the transceiver.

Attenuation: The loss of a signal's energy due to the resistance of the medium.

Authentication Header (AH) Protocol: A protocol defined by IPSec at the network layer that provides integrity to a message through the creation of a digital signature by a hashing function.

Authentication server (AS): The KDC in the Kerberos protocol.

Authentication: Verification of the sender of a message.

Automatic repeat request (ARQ): An error-control method in which correction is made by retransmission of data.

Autonomous system (AS): A group of networks and routers under the authority of a single administration.

Available bit rate (ABR): The minimum data rate in ATM at which cells can be delivered.

Back off: In multiple access, waiting before re-sending after a collision.

Backbone router: A router inside the backbone.

Backbone A network that connects smaller networks in an organization.

Bandwidth The difference between the highest and the lowest frequencies of a composite signal. It also measures the information-carrying capacity of a line or a network.

Bipolar encoding A digital-to-digital encoding method in which 0 amplitude represents binary 0 and positive and negative amplitudes represent alternate 1s.

Bit binary digit; the smallest unit of data(0 or 1)

Bit interval The time required to send one bit.

Bit padding In TDM, the addition of extra bits to a device's source stream to force speed relationships.

Bit rate The number of bits transmitted per second.

Bit stuffing In a bit-oriented protocol, the process of adding an extra bit in the data section of a frame to prevent a sequence of bits from looking like a flag.

Bit-oriented protocol A protocol in which the data frame is interpreted as a sequence of bits.

Bits per second (bps) A measurement of data speed; bits transmitted per second.

Block cipher An encryptiondecryption algorithm that has a block of bits as its basic unit.

Bluetooth A wireless LAN technology designed to connect devices of different functions such as telephones and notebooks in a small area such as a room.

BNC connector A common coaxial cable connector.

Border Gateway Protocol (BGP) An interautonomous system routing protocol based on path vector routing.

Bridge A network device operating at the first two layers of the Internet model with filtering and forwarding capabilities.

Broadband transmission Transmission of signals using modulation of a higher frequency signal. The term implies a wide-bandwidth data combined from different sources.

Broadcast address An address that allows transmission of a message to all nodes of a network.

Broadcasting Transmission of a message to all nodes in a network.

Browser An application program that displays a WWW document. A browser usually uses other Internet services to access the document.

Burst error Error in a data unit in which two or more bits have been altered.

Bursty data Data with varying instantaneous transmission rates.

Bus topology A network topology in which all computers are attached to a shared medium.

Byte A group of eight bits.

Byte stuffing In a byte-oriented protocol, the process of adding an extra byte in the data section of a frame to prevent a byte from looking like a flag.

Byte-oriented protocol A protocol in which the data section of the frame is interpreted as a sequence of bytes (characters).

Caesar cipher A shift cipher used by Julius Caesar with the key value of 3.

Carrier sense multiple access (CSMA) A contention access method in which each station listens to the line before transmitting data.

Carrier sense multiple access with collision avoidance (CSMACA) An access method in which collision is avoided.

Carrier sense multiple access with collision detection (CSMACD) An access method in which stations transmit whenever the transmission medium is available and retransmit when collision occurs.

Carrier signal A high frequency signal used for digital-to-analog or analog-to-analog modulation. One of the characteristics of the carrier signal (amplitude, frequency, or phase) is changed according to the modulating data.

Certification Authority (CA) An agency such as a federal or state organization that binds a public key to an entity and issues a certificate.

Channel A communications pathway.

Checksum A field used for error detection. It is formed by adding bit streams using one's complement arithmetic and then complementing the result.

Cipher An encryptiondecryption algorithm.

Cipher block chaining (CBC) mode A DES and triple DES operation mode in which the encryption (or decryption) of a block depends on all previous blocks.

Cipher feedback mode (CFM) A DES and triple DES operation mode in which data is sent and received 1 bit at a time, with each bit independent of the previous bits.

Cipher stream mode (CSM) A DES and triple DES operation mode in which data is sent and received 1 byte at a time.

Cipher suite AA list of possible ciphers.

Ciphertext The encrypted data.

Circuit switching A switching technology that establishes an electrical connection between stations using a dedicated path.

Cladding Glass or plastic surrounding the core of an optical fiber; the optical density of the cladding must be less than that of the core.

Class A address An IPv4 address with the first octet between 0 and 127.

Class B address An IPv4 address with the first octet between 128 and 191.

Class C address An IPv4 address with the first octet between 192 and 223.

Class D address An IPv4 multicast address.

Class E address An IPv4 address reserved for special purposes.

Classful addressing An IPv4 addressing mechanism in which the IP address space is divided into 5 classes: A, B, C, D, and E. Each class occupies some part of the whole address space.

Client process A running application program on a local site that requests service from a running application program on a remote site.

Client-server model The model of interaction between two application programs in which a program at one end (client) requests a service from a program at the other end (server).

Coaxial cable A transmission medium consisting of a conducting core, insulating material, and a second conducting sheath.

Code division multiple access (CDMA) A multiple access method in which one channel carries all transmissions simultaneously.

Codeword The encoded dataword.

Collision The event that occurs when two transmitters send at the same time on a channel designed for only one transmission at a time; data will be destroyed.

Congestion avoidance In Frame Relay, a method using two bits that explicitly notify the source and destination of congestion.

Congestion control A method to manage network and internetwork traffic to improve throughput.

Congestion Excessive network or internetwork traffic causing a general degradation of service.

Connecting device A tool that connects computers or networks.

Connection control The technique used by the transport layer to deliver segments.

Connection establishment The preliminary setup necessary for a logical connection prior to actual data transfer.

Connection termination A message sent to end a connection.

Connectionless service A service for data transfer without connection establishment or termination.

Connection-oriented concurrent server A connection-oriented server that can serve many clients at the same time.

Connection-oriented service A service for data transfer involving establishment and termination of a connection.

Constant bit rate (CBR) The data rate of an ATM service class that is designed for customers requiring real-time audio or video services.

Consultative Committee for International Telegraphy and Telephony (CCITT) An international standards group now known as the ITU-T.

Contention An access method in which two or more devices try to transmit at the same time on the same channel.

Controlled access A multiple access method in which the stations consult one another to determine who has the right to send.

Core-Based Tree (CBT) In multicasting, a group-shared protocol that uses a center router as the root of the tree.

CRC checker The process that validates the CRC remainder.

CRC generator The process that creates the CRC remainder.

Critical angle In refraction, the value of the angle of incidence that produces a 90-degree angle of refraction.

Crosstalk The noise on a line caused by signals traveling along another line.

Cryptography The science and art of transforming messages to make them secure and immune to attacks.

Cyclic code A linear cod in which the cyclic shifting (rotation) of each codeword creates another code word.

CSNET A network sponsored by the National Science Foundation originally intended for universities.

Cycle The repetitive unit of a periodic signal.

Cyclic redundancy check (CRC) A highly accurate error-detection method based on interpreting a pattern of bits as a polynomial.

Data connection The FTP connection used for data transfer.

Data encryption standard (DES) The U.S. government standard encryption method for nonmilitary and nonclassified use.

Datagram approach (to packet switching) A data transmission method in which each data unit is independent of others.

Datagram In packet switching, an independent data unit.

Datagram network A packet-switched network in which packets are independent from each other.

Decibel (dB) A measure of the relative strength of two signal points.

Decryption Recovery of the original message from the encrypted data.

Demodulation The process of separating the carrier signal from the information-bearing signal.

Differential Manchester encoding A digital-to-digital polar encoding method that features a transition at the middle of the bit interval as well as an inversion at the beginning of each 1 bit.

Distance vector routing A routing method in which each router sends its neighbors a list of networks it can reach and the distance to that network.

Distortion Any change in a signal due to noise, attenuation, or other influences.

Domain Name System (DNS) A TCPIP application service that converts user-friendly names to IP addresses.

Electromagnetic spectrum The frequency range occupied by electromagnetic energy.

Electronic code block (ECB) mode A DES and triple DES operation method in which a long message is divided into 64-bit blocks before being encrypted separately.

Electronic mail (email) A method of sending messages electronically based on mailbox addresses rather than a direct host-to-host exchange.

Electronics Industries Association (EIA) An organization that promotes electronics manufacturing concerns. It has developed interface standards such as EIA-232, EIA-449, and EIA-530.

Encapsulation The technique in which a data unit from one protocol is placed within the data field portion of the data unit of another protocol.

Encryption Converting a message into an unintelligible form that is unreadable unless decrypted.

Error control The detection and handling of errors in data transmission.

Error correction by retransmission The process of correcting bits by resending the data.

Even parity An error-detection method in which an extra bit is added to the data unit so that the total number of 1s becomes even.

Fast Ethernet Ethernet with a data rate of 100 Mbps.

Fast retransmission Retransmission of a segment in TCP protocol when three acknowledgements have been received that imply the loss or corruption of that segment.

Federal Communications Commission (FCC) A government agency that regulates radio, television, and telecommunications.

Fiber distributed data interface (FDDI) A high-speed (100-Mbps) LAN, defined by ANSI, using fiber optics, dual ring topology, and the token-passing access method. Today an FDDI network is also used as a MAN.

File Transfer Protocol (FTP) In TCPIP, an application layer protocol that transfers files between two sites.

Filtering A process in which a bridge makes forwarding decisions.

Finite state machine A machine that goes through a limited number of states.

Firewall A device (usually a router) installed between the internal network of an organization and the rest of the Internet to provide security.

Flooding Saturation of a network with a message.

Flow control A technique to control the rate of flow of frames (packets or messages).

Forward error correction Correction of errors at the receiver.

Fragmentation offset A field in the IP header used in fragmentation to show the relative position of the fragment with respect to the whole datagram.

Fragmentation The division of a packet into smaller units to accommodate a protocol's MTU.

Frame A group of bits representing a block of data.

Frame bursting A technique in CSMACD Gigabit Ethernet in which multiple frames are logically connected to each other to resemble a longer frame.

Frame check sequence (FCS) The HDLC error-detection field containing either a 2- or 4-byte CRC.

Frame Relay A packet-switching specification defined for the first two layers of the Internet model. There is no network layer. Error checking is done on end-to-end basis instead of on each link.

Gateway A device used to connect two separate networks that use different communication protocols.

Gigabit Ethernet Ethernet with a 1000 Mbps data rate.

Global Internet The Internet.

Global Positioning System (GPS) An MEO public satellite system consisting of 24 satellites and used for land and sea navigation. GPS is not used for communications.

Global System for Mobile Communication (GSM) A second-generation cellular phone system used in Europe.

Go-Back-N ARQ An error-control method in which the frame in error and all following frames must be retransmitted.

Ground propagation Propagation of radio waves through the lowest portion of the atmosphere (hugging the earth).

Half-duplex mode A transmission mode in which communication can be two-way but not at the same time.

Hamming code A method that adds redundant bits to a data unit to detect and correct bit errors.

Hamming distance The number of differences between the corresponding bits in two datawords.

Handoff Changing to a new channel as a mobile device moves from one cell to another.

Handshake protocol A protocol to establish or terminate a connection.

Header Control information added to the beginning of a data packet. Also, in an email, the part of the message that defines the sender, the receiver, the subject of the message, and other information.

Header translation Conversion of the IPv6 header to IPv4.

Hierarchical routing A routing technique in which the entire address space is divided into levels based on specific criteria.

High-level Data Link Control (HDLC) A bit-oriented data link protocol defined by the ISO. It is used in X.25 protocol. A subset, called link access procedure (LAP), is used in other protocols. It is also a base for many data link protocols used in LANs.

Hop count The number of nodes along a route. It is a measurement of distance in routing algorithms.

Hop limit An IPv6 field that limits the number of routers that a packet can visit.

Hop-to-hop delivery Transmission of frames from one node to the next. table.

Hub A central device in a star topology that provides a common connection among the nodes.

Huffman encoding A statistical compression method using variable-length codes to encode a set of symbols.

HyperText Markup Language (HTML) The computer language for specifying the contents and format of a web document. It allows additional text to include codes that define fonts, layouts, embedded graphics, and hypertext links.

HyperText Transfer Protocol (HTTP) An application service for retrieving a web document.

Institute of Electrical and Electronics Engineers (IEEE) A group consisting of professional engineers which has specialized societies whose committees prepare standards in members' areas of specialty.

Integrity A data quality of being noncorrupted.

Interior routing Routing inside an autonomous system.

Interleaving Taking a specific amount of data from each device in a regular order.

International Organization of Standardization (ISO) A worldwide organization that defines and develops standards on a variety of topics.

International Telecommunications Union-Telecommunication Standardization Sector (ITU-T) A standards organization formerly known as the CCITT.
internet A collection of networks connected by internetworking devices such as routers or gateways.

Internet A global internet that uses the TCPIP protocol suite.

Internet address A 32-bit or 128-bit network-layer address used to uniquely define a host on an internet using the TCPIP protocol.

Internet Architecture Board (IAB) The technical adviser to the ISOC; oversees the continuing development of the TCPIP protocol suite.

Internet Assigned Numbers Authority (IANA) A group supported by the U.S. government that was responsible for the management of Internet domain names and addresses until October 1998.

Internet Control Message Protocol (ICMP) A protocol in the TCPIP protocol suite that handles error and control messages.

Internet Control Message Protocol, version 6 (ICMPv6) A protocol in IPv6 that handles error and control messages.

Internet Corporation for Assigned Names and Numbers (ICANN) A private, nonprofit corporation managed by an international board that assumed IANA operations.
Internet draft A working Internet document (a work in progress) with no official status and a six-month lifetime.

Internet Engineering Steering Group (IESG) An organization that oversees the activity of IETF.

Internet Engineering Task Force (IETF) A group working on the design and development of the TCPIP protocol suite and the Internet.

Internet Group Management Protocol (IGMP) A protocol in the TCPIP protocol suite that handles multicasting.

Internet Protocol version 4 (IPv4) The current version of Internet Protocol.

Internet Protocol, version 6 (IPv6) The sixth version of the Internetworking Protocol; it features major IP addressing changes.

Internet Research Task Force (IRTF) A forum of working groups focusing on long-term research topics related to the Internet.

Internetwork (internet) A network of networks.

Internetworking Connecting several networks together using internetworking devices such as routers and gateways.

Intranet A private network that uses the TCPIP protocol suite.

IP datagram The Internetworking Protocol data unit.

Jitter A phenomenon in real-time traffic caused by gaps between consecutive packets at the receiver.

Joint Photographic Experts Group (JPEG) A standard for compressing continuous-tone picture.

Keepalive message A message that establishes a relationship between the two routers.

Keepalive timer A timer that prevents a long idle connection between two TCPs.

Kerberos An authentication protocol used by Windows 2000.

Key A number that a cipher operates on.

Key distribution center (KDC) In secret key encryption, a trusted third party that shares a key with each user.

Layered architecture A model based on ordered tiers.

Leaky bucket algorithm An algorithm to shape bursty traffic.

Least-cost tree An MOSPF feature in which the tree is based on a chosen metric instead of shortest path.

Line coding Converting binary data into signals.

Local address The part of an email address that defines the name of a special file, called the user mailbox, where all of the mail received for a user is stored for retrieval by the user agent.

Local area network (LAN) A network connecting devices inside a single building or inside buildings close to each other.

Logical address An address defined in the network layer.

Logical link control (LLC) The upper sublayer of the data link layer as defined by IEEE Project 802.2.

Medium access control (MAC) sublayer The lower sublayer in the data link layer defined by the IEEE 802 project. It defines the access method and access control in different local area network protocols.

Metropolitan area network (MAN) A network that can span a geographical area the size of a city.

Motion picture experts group (MPEG) A method to compress videos.

multicast address An address used for multicasting.

Multicast router A router with a list of loyal members related to each router interface that distributes the multicast packets.

Multicasting A transmission method that allows copies of a single packet to be sent to a selected group of receivers.

Multipurpose Internet Mail Extension (MIME) A supplement to SMTP that allows non-ASCII data to be sent through SMTP.

Multistage switch An array of switches designed to reduce the number of crosspoints.

Nagle's algorithm An algorithm that attempts to prevent silly window syndrome at the sender's site; both the rate of data production and the network speed are taken into account.

Name space All the names assigned to machines on an internet.

Negative acknowledgment (NAK) A message sent to indicate the rejection of received data.

Network A system consisting of connected nodes made to share data, hardware, and software.

Nnetwork access point (NAP) A complex switching station that connects backbone networks.

Network address An address that identifies a network to the rest of the Internet; it is the first address in a block.

Network Control Protocol (NCP) In PPP, a set of control protocols that allows the encapsulation of data coming from network layer protocols.

Network interface card (NIC) An electronic device, internal or external to a station, that contains circuitry to enable the station to be connected to the network.

Network layer The third layer in the Internet model, responsible for the delivery of a packet to the final destination.

Nyquist theorem A theorem that states that the number of samples needed to adequately represent an analog signal is equal to twice the highest frequency of the original signal.

Odd parity An error-detection method in which an extra bit is added to the data unit such that the sum of all 1-bits becomes odd.

Open shortest path first (OSPF) An interior routing protocol based on link state routing.

Open system A model that allows two different systems to communicate regardless of their underlying architecture.

Open Systems Interconnection (OSI) model A seven-layer model for data communication defined by ISO.

Packet switching Data transmission using a packet-switched network.

Packet Synonym for data unit, mostly used in the network layer.

Packet-switched network A network in which data are transmitted in independent units called packets.

Parallel transmission Transmission in which bits in a group are sent simultaneously, each using a separate link.

Parity bit A redundant bit added to a data unit (usually a character) for error checking.

Parity check An error-detection method using a parity bit.

Password Authentication Protocol (PAP) A simple two-step authentication protocol used in PPP.

Path layer A SONET layer responsible for the movement of a signal from its optical source to its optical destination.

Permanent virtual circuit (PVC) A virtual circuit transmission method in which the same virtual circuit is used between source and destination on a continual basis.

Persistent strategy In CSMA, a strategy in which the station sends a frame after sensing the line.

Physical address The address of a device used at the data link layer (MAC address).

Physical layer The first layer of the Internet model, responsible for the mechanical and electrical specifications of the medium.

Physical topology The manner in which devices are connected in a network.

Piggybacking The inclusion of acknowledgment on a data frame.

Pipelining In Go-Back-*n* ARQ, sending several frames before news is received concerning previous frames.

P-persistent A CSMA persistence strategy in which a station sends with probablity p if it finds the line idle.

P-persistent strategy A CSMA persistence strategy in which a station sends with probability *p* if it finds the line idle.

Preamble The 7-byte field of an IEEE 802.3 frame consisting of alternating 1s and 0s that alert and synchronize the receiver.

Propagation speed The rate at which a signal or bit travels; measured by distancesecond.

Protocol Rules for communication.

Proxy server A computer that keeps copies of responses to recent requests.

Pruning Stopping the sending of multicast messages from an interface.

Pseudoheader Information from the IP header used only for checksum calculation in UDP and TCP packet.

Public-key cryptography A method of encryption based on a nonreversible encryption algorithm. The method uses two types of keys: The public key is known to the public; the private key (secret key) is known only to the receiver.

Pure ALOHA The original ALOHA.

Pulse rate The number of symbols per second.

Quality of service (QoS) A set of attributes related to the performance of the connection.

Quantization The assignment of a specific range of values to signal amplitudes.

Remote access Using a terminal that is not directly connected to a computer.

Remote bridge A device that connects LANs and point-to-point networks; often used in a backbone network.

Rivest, Shamir, Adleman (RSA) encryption See *RSA encryption*.

RJ45 A coaxial cable connector.

Round-trip time (RTT) The time required for a datagram to go from a source to a destination and then back again.

Route A path traveled by a packet.

Router An internetworking device operating at the first three OSI layers. A router is attached to two or more networks and forwards packets from one network to another.

Router link LSA An LSA packet that advertises all of the links of a router.

Routing table A table containing information a router needs to route packets. The information may include the network address, the cost, the address of the next hop, and so on.

Routing The process performed by a router; finding the next hop for a datagram.

RSA encryption A popular public-key encryption method developed by Rivest, Shamir, and Adleman.

Sampling rate The number of samples obtained per second in the sampling process.

Sampling The process of obtaining amplitudes of a signal at regular intervals.

S-box An encryption device made of decoders, P-boxes, and encoders.

Secret-key encryption A security method in which the key for encryption is the same as the key for decryption; both sender and receiver have the same key.

Security The protection of a network from unauthorized access, viruses, and catastrophe.

Segment The packet at the TCP layer. Also, the length of transmission medium shared by devices.

Segmentation and reassembly (SAR) The lower AAL sublayer in the ATM protocol in which a header andor trailer may be added to produce a 48-byte element.

Segmentation The splitting of a message into multiple packets; usually performed at the transport layer.

Selective-repeat ARQ An error-control method in which only the frame in error is resent.

Session layer The fifth layer of the OSI model, responsible for the establishment, management, and termination of logical connections between two end users.

Setup phase In virtual circuit switching, a phase in which the source and destination use their global addresses to help switches make table entries for the connection.

S-frame An HDLC frame used for supervisory functions such as acknowledgment, flow control, and error control; it contains no user data.

Simple Mail Transfer Protocol (SMTP) The TCPIP protocol defining electronic mail service on the Internet.

Simple Network Management Protocol (SNMP) The TCPIP protocol that specifies the process of management in the Internet.

Simplex mode A transmission mode in which communication is one way.

Sliding window A protocol that allows several data units to be in transition before receiving an acknowledgment.

Sliding window ARQ An error-control protocol using sliding window concept.

Slotted ALOHA The modified ALOHA access method in which time is divided into slots and each station is forced to start sending data only at the beginning of the slot.

Socket address A structure holding an IP address and a port number.

Socket An end point for a process; two sockets are needed for communication.

Socket interface An API based on UNIX that defines a set of system calls (procedures) that are an extension of system calls used in UNIX to access files.

Source routing Explicitly defining the route of a packet by the sender of the packet.

Source-based tree A tree used for multicasting by multicasting protocols in which a single tree is made for each combination of source and group.

Source quench message An ICMP message sent to slow down or stop the sending of datagrams.

Source-to-destination delivery The transmission of a message from the original sender to the intended recipient.

Space-division switching Switching in which the paths are separated from each other spatially.

Spanning tree A tree with the source as the root and group members as leaves; a tree that connects all of the nodes.

Spanning tree algorithm An algorithm that prevents looping when two LANs are connected by more than one bridge.

Static routing A type of routing in which the routing table remains unchanged.

Stop bit In asynchronous transmission, one or more bits to indicate the end of transmission.

Stop-and-wait ARQ An error-control protocol using stop-and-wait flow control.

Stop-and-Wait Protocol A protocol in which the sender sends on frame, stops until it receives confirmation from the receiver, and then sends the next frame.

Subnet subnetwork.

Subnet address The network address of a subnet.

Subnet mask The mask for a subnet.

Switch A device connecting multiple communication lines together.

Switched Ethernet An Ethernet in which a switch, replacing the hub, can direct a transmission to its destination.

Switched virtual circuit (SVC) A virtual circuit transmission method in which a virtual circuit is created and in existence only for the duration of the exchange.

Symmetric key The key used for both encryption and decryption.

Symmetric-key cryptography A cipher in which the same key is used for encryption and decryption.

Synchronous Optical Network (SONET) A standard developed by ANSI for fiber optic technology that can transmit high-speed data. It can be used to deliver text, audio, and video.

TCP timer The timers used by TCP to handle retransmission, zero window-size advertisements, long idle connections, and connection termination.

TCPIP protocol suite A group of hierarchical protocols used in an internet.

Teleconferencing Audio and visual communication between remote users.

Terminal Network (TELNET) A general purpose client-server program that allows remote login.

Three-way handshaking A sequence of events for connection establishment or termination consisting of the request, then the acknowledgment of the request, and then confirmation of the acknowledgment.

Terminating state A PPP state in which several packets are exchanged between the two ends for house cleaning and closing the link.

Three-way handshake A sequence of events for connection establishment or termination consisting of the request, then the acknowledgment of the request, and then confirmation of the acknowledgment.

Throughput The number of bits that can pass through a point in one second.

Time to live (TTL) The lifetime of a packet.

Time-division multiplexing (TDM) The technique of combining signals coming from low-speed channels to share time on a high-speed path.

Token A small packet used in token-passing access method.

Token bucket An algorithm that allows idle hosts to accumulate credit for the future in the form of tokens.

Token passing An access method in which a token is circulated in the network. The station that captures the token can send data.

Token Ring A LAN using a ring topology and token-passing access method.

Topology The structure of a network including physical arrangement of devices.

Traffic control A method for shaping and controlling traffic in a wide area network.

Traffic shaping A mechanism to control the amount and the rate of the traffic sent to the network to improve QoS.

Trailer Control information appended to a data unit.

Transmission Control Protocol (TCP) A transport protocol in the TCPIP protocol suite.

Transmission Control ProtocolInternetworking Protocol (TCPIP) A five-layer protocol suite that defines the exchange of transmissions across the Internet.

Transmission rate The number of bits sent per second.

Transparency The ability to send any bit pattern as data without it being mistaken for control bits.

Transparent bridge Another name for a learning bridge.

Transparent data Data that can contain control bit patterns without being interpreted as control.

Transport Layer Security (TLS) A security protocol at the transport level designed to provide security on the WWW.

Transport layer The fourth layer in the Internet and OSI model; responsible for reliable end-to-end delivery and error recovery.

Tunneling In multicasting, a process in which the multicast packet is encapsulated in a unicast packet and then sent through the network. In VPN, the encapsulation of an encrypted IP datagram in a second outer datagram. For IPv6, a strategy used when two computers using IPv6 want to communicate with each other when the packet must pass through a region that uses IPv4.

Unguided medium A transmission medium with no physical boundaries.

Unicast address An address belonging to one destination.

Unicast message A message sent to just one destination.

Unicast routing The sending of a packet to just one destination.

Unicasting The sending of a packet to just one destination.

Unicode The international character set used to define valid characters in computer science.

User Datagram Protocol (UDP) A connectionless TCPIP transport layer protocol.

User datagram The name of the packet in the UDP protocol.

V series ITU-T standards that define data transmission over telephone lines. Some common standards are V.32, V32bis, V.90, and V92.

Variable bit rate (VBR) The data rate of an ATM service class for users needing a varying bit rate.

Video Recording or transmitting of a picture or a movie.

Video band In an HFC network, the band from 54 to 550 MHz for downstream video.

Virtual channel identifier (VCI) A field in an ATM cell header that defines a channel.

Virtual circuit (VC) A logical circuit made between the sending and receiving computer. The connection is made after both computers do handshaking. After the connection, all packets follow the same route and arrive in sequence.

Virtual circuit approach to packet switching A packet switching method in which all packets of a message or session follow the exact same route.

Virtual circuit switching A switching technique used in switched WANs.

Virtual path (VP) In ATM, a connection or set of connections between two switches.

Voice over IP A technology in which the Internet is used as a telephone network.

Web page A unit of hypertext or hypermedia available on the Web.

Web Synonym for World Wide Web (WWW).

Well-known port number A port number that identifies a process on the server.

Well-known port A port number that identifies a process on the server.

Wide area network (WAN) A network that uses a technology that can span a large geographical distance.

Wireless communication Data transmission using unguided media.

Wireless LAN A LAN which uses unguided media.

World Wide Web (WWW) A multimedia Internet service that allows users to traverse the Internet by moving from one document to another via links that connect them together.

X.25 An ITU-T standard that defines the interface between a data terminal device and a packet-switching network.

Zone In DNS, what a server is responsible for or has authority over.