

CONTROL AT THE EDGE OR INSIDE THE NETWORK?

The Internet and the traditional telephone network are fundamentally different in their approach to service control. The Internet has always placed control in the end system and made minimal assumptions about the network—in fact, the network of networks. The telephone network has always placed service control in the switches inside the network. The two approaches lead to fundamentally different philosophies with respect to the introduction of new services. The traditional Internet approach, by keeping control in the end system, allows any new application that operates over TCP/IP to be readily introduced into the network, without the permission of the operator. The traditional telephone approach by keeping control inside the network allows the network operator to decide what services to deploy in the network. The experience with the World Wide Web has demonstrated that the Internet approach leads to a much faster rate of service introduction. This is, of course, due to the fact that the Internet empowers a much larger community to define and introduce services. Nevertheless, it is possible that, as the Internet service industry matures and a few dominant operators emerge, there will be a temptation to control the services that are available to the end users by progressively moving service control back inside the network. We can only hope that network architectures will emerge that will allow network-centric and network-edge control protocols to co-exist and thus foster competition and continue to allow end users to innovate and to drive the introduction of new services.

SUMMARY

In this chapter we have examined various network architectures that were designed to improve the performance and value of the current Internet infrastructure. We described the integrated services model that provides per flow QoS in the Internet environment, and the associated signaling protocol called RSVP. We also examined the differentiated services model that provides a simpler form of QoS in a more scalable solution.


We considered the peer model in which IP routing and addressing are combined with layer 2 forwarding. In particular we introduced MPLS, which combines network layer routing with layer 2 forwarding to produce a system that combines low-cost, very high scalability, and flexibility in the routing paradigms that can be supported. We also discussed how the signaling and routing capabilities of MPLS are generalized in GMPLS to make circuit-based SONET and optical networks more responsive, flexible, and efficient.

Finally, we examined several protocols that are intended to facilitate the deployment of multimedia services in the Internet. RTP and RTCP provide the means for encapsulating many types of multimedia information over UDP. SIP and H.323 complement RTP by providing a means for setting up, maintaining, and terminating multimedia sessions.

CHECKLIST OF IMPORTANT TERMS

| | |
|--|--------------------------------------|
| Assured Forwarding PHB (AF PHB) | overlay model |
| bandwidth broker | path |
| controlled-load service | path protection/restoration |
| differentiated services (diffserv) model | PathTear |
| dropper | peer model |
| Expedited Forwarding PHB (EF PHB) | per-hop behavior (PHB) |
| filterspec | protection |
| flowspec | protection path |
| Forwarding Equivalence Class (FEC) | Real-Time Transport Protocol (RTP) |
| Generalized MPLS (GMPLS) | ReSerVation Protocol (RSVP) |
| global repair | restoration |
| guaranteed service | Resv |
| integrated services (intserv) model | ResvTear |
| Label Distribution Protocol (LDP) | RSVP-TE |
| Label Switching Router (LSR) | RTP Control Protocol (RTCP) |
| Label Switched Path (LSP) | service level agreement (SLA) |
| link bundling | session |
| Link Management Protocol (LMP) | shaper |
| link protection/restoration | Session Initiation Protocol (SIP) |
| LSP hierarchy | traffic classifier |
| local repair | traffic conditioner |
| marker | traffic conditioning agreement (TCA) |
| meter | VC merging |
| MultiProtocol Label Switching (MPLS) | virtual private network (VPN) |
| Multiprotocol Over ATM (MPOA) | working path |
| node protection/restoration | |

FURTHER READING

-  ATM Forum Technical Committee, "Multi-Protocol Over ATM Version 1 Specification," July 1997.
- Clark, D. and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE Transactions on Networking*, August 1998.
- Davie, B., P. Doolan, and Y. Rekhter, *Switching in IP Networks*, Morgan Kaufman, 1998.
- Floyd, S. and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE Transactions on Networking*, August 1993.
- Huitema, C., J. Cameron, P. Mouchteris, and D. Smyk, "An Architecture for Residential Internet Telephony Service," *IEEE Network*, Vol. 13, No. 3, pp. 50–56.
- Ibe, O., *Essentials of ATM Networks and Services*, Addison-Wesley, Reading, Massachusetts, 1997.
- Newman, P., G. Minshall, T. Lyon, and L. Huston, "IP Switching and Gigabit Routers," *IEEE Communications Magazine*, pp. 64–69, January 1997.

- Schulzrinne, H. and J. Rosenberg, "A Comparison of SIP and H.323 for Internet Telephony," *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Cambridge, England), July 1998.
- Schulzrinne, H. and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Services across the Internet," *Bell Labs Technical Journal*, Vol. 3, pp. 144–160, October–December 1998.
- Schulzrinne, H. and J. Rosenberg, "The IETF Internet Telephony Architecture and Protocols," *IEEE Network*, Vol. 13, No. 3, pp. 18–23, 1999.
- Widjaja, I. and A. I. Elwalid, "Performance Issues in VC-Merge Capable Switches for Multiprotocol Label Switching," *IEEE Journal in Selected Area on Communications*, Vol. 17, No. 6, pp. 1178–1189, June 1999.
- RFC 1889, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "A Transport Protocol for Real-Time Applications," January 1996.
- RFC 1890, H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," January 1996.
- RFC 2205, R. Braden et al., "Resource Reservation Protocol (RSVP)," September 1997.
- RFC 2211, J. Wroclawski, "Specification of a Controlled-Load Network Element Service," September 1997.
- RFC 2212, S. Shenker et al., "Specification of Guaranteed Quality of Service," September 1997.
- RFC 2250, D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video," January 1998.
- RFC 2430, T. Li and Y. Rekhtor, "A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," October 1998.
- RFC 2475, S. Blake et al., "An Architecture for Differentiated Services," December 1998.
- RFC 2543, M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "IP: Session Initiation Protocol," March 1999.
- RFC 3015, F. Cuervo, N. Greene, C. Huitema, A. Rayhan, B. Rosen, and J. Segers, "Megaco Protocol Version 1.0," November 2000.
- RFC 3031, E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," January 2001.
- RFC 3036, L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP Specification," January 2001.
- RFC 3209, D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," December 2001.

See our website for additional references available through the Internet.

PROBLEMS

- 10.1. Can all traffic in an intServ router be of guaranteed service type? Explain.
- 10.2. Develop a packet-scheduling approach to provide guaranteed service, controlled load service, and best-effort service in a router. Can any mix of these three traffic types be supported? Explain.
- 10.3. RSVP signaling is very different from ATM signaling. Discuss the differences and list the advantages and disadvantages of each protocol.
- 10.4. Consider the merging of reservations in RSVP. Suppose that there is a single sender and 2^n receivers that are reached through path that consists of a depth n binary tree that has

router located at each node. What is the total number of reservation request messages that are generated by the receivers?

- 10.5.** Suppose RSVP is used to set up a virtual private network to interconnect n users across an Internet. Discuss the appropriateness of the three RSVP reservation styles for this application.
- 10.6.** Select and justify a choice an RSVP reservation style that is appropriate for the following situations:
- (a) A multiuser application where a video stream from each participant must be delivered to every other participant.
 - (b) An audioconferencing application that is self-limiting in that more than two participants are unlikely to speak simultaneously.
 - (c) A multiuser application in which each participant must receive explicit permission before they can transmit information.
- 10.7.** What information needs to be carried by Resv messages in RSVP to set up guaranteed IP service with a specific bandwidth and a given delay bound requirement? How does a router process this information?
- 10.8.** Discuss the interplay between QoS routing algorithm, RSVP, and integrated services IP.
- 10.9.** Explain how the soft-state feature of RSVP allows it to adapt to failures in the network.
- 10.10.** Discuss whether RSVP provides a scalable means to set up connections to a session that consists of the broadcasting of a given TV program.
- 10.11.** Explain what happens if RSVP is used to set up small bandwidth flows on a large bandwidth link? How well does RSVP scale in this situation?
- 10.12.** Obtain a packet capture file for RSVP either in the lab or downloaded from a website. Examine the details of Path, Resv, and refresh messages.
- 10.13.** What RSVP reservation style is appropriate for distributed network gaming?
- 10.14.** What RSVP reservation style is appropriate for instant messaging? For stock price notification?
- 10.15.** A sender transmits layered video information. Provide appropriate reservation messages for the receivers in Figure 10.5 if Rx1 is on a 802.11 wireless LAN, Rx2 is on a 3G cell phone, and Rx3 is a workstation on a fast Ethernet LAN.
- 10.16.** Derive the state lifetime L in RSVP.
- 10.17.** Discuss possible scheduling mechanisms (e.g., head-of-line priority, weighted fair queueing) that you could use to implement EF PHB, AF PHB group, and DE PHB. Discuss the trade-offs between performance and complexity.
- 10.18.** Perform a web search of router equipment that implements differentiated services IP. Find out what PHBs are supported by the routers, and if possible, find out what scheduling mechanisms are used to implement the PHBs.

- 10.19.** Perform a web search for microelectronic chips that implement packet scheduling. What QoS mechanisms are supported by these chips? At what bit rates? Do the chips require external memory and if so how much? Explain the memory requirements.
- 10.20.** Provide arguments for and against the statement: "It is essential that Ethernet provide QoS."
- 10.21.** Can MPLS make use of DS information to determine the forwarding experienced by different traffic flows? If yes, explain how. If no, explain why not.
- 10.22.** What is an appropriate traffic conditioner for EF PHB packets? AF PHB packets?
- 10.23.** Suppose that two RSVP-capable users are each connected to an intranet which in turn connect them through an Internet backbone that provides differentiated services IP. Suppose that the intranets provide integrated services IP. Explain the functions that must be implemented in the intranet routers in order to provide controlled load service and guaranteed service.
- 10.24.** Compare integrated services IP and differentiated services IP in terms of their ability to protect themselves against theft of service.
- 10.25.** Explain why shaping may be required at the egress node of a differentiated services domain.
- 10.26.** With reference to Figure 10.12, explain how the various types of routers are configured to support packet transport services that require end-to-end low delay and low jitter (i.e., voice over IP).
- 10.27.** A common IP-over-ATM overlay model uses an ATM permanent virtual connection (PVC) to connect each pair of IP routers thereby creating a fully meshed topologies for the IP network. Thus $N(N - 1)/2$ PVCs are needed if there are N routers. Suppose that the routers run a link-state protocol (e.g., OSPF) among themselves. How many advertisements would the routers receive if a given PVC is down? Repeat if a given router is down.
- 10.28.** Discuss the advantages and disadvantages of different label assignment approaches (i.e., traffic-driven, topology-driven, and request-driven) in terms of number of connections, and latency.
- 10.29.** MPOA uses flow detection to set up virtual circuits whereas MPLS relies on routing topology to establish virtual circuits. Discuss the advantages and disadvantages of the two approaches.
- 10.30.** Consider explicit routing in MPLS.
- Can explicit routing in MPLS be used to select a route that meets the QoS requirements of a particular flow (e.g., guaranteed bandwidth and delay)? Explain.
 - Give an example at a fine level of granularity that can make use of the QoS explicit routing in part (a). Give another example at a coarser level of granularity.
 - Compare the case where the first LSR computes an explicit path to the case where all LSRs participate in the selection of the path. What information does the first LSR require to select an explicit path?

- 10.31.** In an MPLS domain, rank the following three flows in terms of their level of aggregation:
 (a) all packets destined to the same host, (b) all packets with the same egress router,
 (c) all packets with the same CIDR address.
- 10.32.** Consider MPLS for unicast traffic. Explain what information is used to specify labels when labels are determined according to (a) host pairs, (b) network pairs, (c) destination network, (d) egress router, (e) next-hop AS, (f) destination AS.
- 10.33.** Explain why it may be useful for the MPLS header to carry a time-to-live value.
- 10.34.** Compare the overall scalability of the following three cases:
 (a) Layer 3 forwarding only: each router performs a longest-prefix match to determine the next forwarding hop.
 (b) Layer 3 forwarding and some Layer 2 MPLS forwarding;
 (c) Layer 2 MPLS forwarding only.
- 10.35.** Are MPLS label switching and packet filtering at firewalls compatible? Explain.
- 10.36.** Discuss what factors determine the level of computational load for label assignment and distribution for the following three cases: (a) topology-driven label assignment, (b) request-driven label assignment, (c) traffic-driven label assignment.
- 10.37.** Compare MPLS label stacks and IP-over-IP encapsulation tunnels in terms of their traffic engineering capabilities.
- 10.38.** Compare the following two approaches to traffic engineering:
 (a) Offline computation: A central site knows the demand (total bandwidth requirement) between each pair of LSRs and computes the paths in the network.
 (b) Online computation: An ingress LSR independently computes a path to a specified egress LSR as new demands arrive at the ingress LSR.
- 10.39.** Suppose that an MPLS network consisting of N routers supports C diffserv traffic classes. A trunk is defined as traffic from a single traffic class that is aggregated into an LSP.
 (a) What is the maximum number of trunks if a trunk is defined for every pair of ingress router and egress router?
 (b) Suppose that the trunks in part (a) are merged if they have the same traffic class and the same egress routers. What is the maximum number of multipoint-to-point trees rooted in an egress router?
 (c) Can you think of other ways of merging trunks?
- 10.40.** Can an MPLS label correspond to all packets that match a particular integrated services filter specification? If yes, explain how such a label may be used. If no, explain why not.
- 10.41.** Suppose that network 2 in Figure 10.15 is a SONET network and that networks 1 and 3 are IP networks. Explain the overlay and peer-to-peer models to network interconnection. What changes if network 1 and network 3 are MPLS networks? What changes if the network 2 uses GMPLS?
- 10.42.** Suppose that network 2 in Figure 10.15 provides optical wavelength service and that networks 1 and 3 are IP networks. Explain the overlay and peer-to-peer models to network

interconnection. What changes if network 1 and network 3 are MPLS networks? What changes if network 2 uses GMPLS?

- 10.43.** Suppose that network 2 in Figure 10.15 provides optical wavelength services and that networks 1 and 3 are SONET networks. Explain the overlay and peer-to-peer models to network interconnection. What changes if all the networks use GMPLS?
- 10.44.** Does the notion of label merging apply to GMPLS? Explain.
- 10.45.** Suppose that MPLS traffic runs over a SONET network. Is protection and restoration required at both the SONET and MPLS levels? Explain.
- 10.46.** Consider a 4-node fully connected optical network. Suppose that each pair of nodes is connected by optical fiber that carries 100 wavelengths in each direction. Compare the scale required to do OSPF routing with and without link bundling.
- 10.47.** Should RTP be a transport layer protocol? Explain why or why not.
- 10.48.** Can RTP be used to provide reliable real-time communication? If yes, explain how. If no, explain why not.
- 10.49.** Can RTP operate over AAL5? Explain.
- 10.50.** Suppose that the marker bit in the RTP header is used to indicate the beginning of a talkspurt. Explain the role of the timestamps and sequence numbers at the receiver.
- 10.51.** Discuss the relative usefulness of RTCP for unicast and multicast connections.
- 10.52.** Run Microsoft NetMeeting or some other conferencing application that carries voice and or video. Capture the packets that are exchanged during the setting up of the session and during the data transfer phase. Determine what protocols are used for session setup and for data transport. Examine details of the fields in each packet type. What type of media encoding is used?
- 10.53.** Repeat the previous problem for a distributed network game.
- 10.54.** Suppose a router acts as a firewall and filters packets. Can the router identify RTP packets? Can the router identify RTP packets from different connections?
- 10.55.** Suppose you want to implement an IP telephony system.
 (a) Look up the RTP payload format for G.711. Discuss the transmission efficiency for the resulting system.
 (b) Repeat for G.729.
- 10.56.** Explain the relevance to scalability of the SIP protocol not having to maintain state. Contrast this with the case of traditional telephone networks.

- 10.57.** Consider whether and how SIP can be used to provide the following services:
- (a) Call Display: The number and/or name of the calling party is listed on a screen before the call is answered.
 - (b) Call Waiting: A special sound is heard when the called party is on the line and another user is trying to reach the called party.
 - (c) Call Answer: If the called party is busy or after the phone rings a prescribed number of times, the network gives the caller the option of leaving a voice message.
 - (d) Three-Way-Calling: Allows a user to talk with two other people at the same time.
- 10.58.** A professor has three locations at his university: his regular professorial office, his lab, and the administrative office for a research center he operates.
- (a) Explain how the *fork* capability of SIP may be used to direct an Internet phone call to the professor.
 - (b) Suppose that the professor spends Tuesdays working at a small startup, where he also has an Internet phone. Explain how a REGISTER message can be used to direct calls from the professorial office in the university to the startup.
 - (c) Suppose that on Wednesday, the professor forgets to change the forwarding instructions from the day before. Can a call placed to him at the startup reach him at the office? At the lab?
- 10.59.** Suppose that a university campus is covered by 802.11 wireless LANs that are connected to the campus IP backbone. Explain how SIP, RTP, and associated protocols can be used to provide mobile phone service across the campus. Explain how the service can be extended to provide service to other universities.
- 10.60.** A recent proposal involves replacing the H.323 gateway with a trunking gateway that is responsible solely for media format conversion. Signaling to establish a call is handled by a call agent that talks to the client on the packet network side, to signaling #7 gateway to the telephone network, and to the trunking gateway.
- (a) Explain how a telephone call is set up between an Internet terminal and a telephone set.
 - (b) Does this system have greater scalability than an H.323 gateway?
 - (c) Can the call agents be used to move telephone network Intelligent Network functionality to the Internet?

Security Protocols

To provide certain services, some communication protocols need to process the information they transmit and receive. For example, protocols that provide reliable communication service encode the transmitted information to detect when transmission errors have occurred so that they can initiate corrective action. Another example is a security protocol that provides a secure communication service that prevents eavesdroppers from reading or altering the contents of messages and prevents imposters from impersonating legitimate users. In this chapter we discuss the following aspects of security:

1. *Security and cryptography.* We introduce the threats that can arise in a network context, and we identify various types of security requirements. We introduce secret key and public key cryptography and explain how they meet the above security requirements.
2. *Security protocols.* We develop protocols that provide security services across insecure networks. We also introduce protocols for establishing a security association and for managing keys. We relate these security protocols to the standard security protocols developed for the IP layer—IP Security (IPSec)—and for the transport layer—Secure Sockets Layer (SSL) and Transport Layer Security (TLS).
3. *Cryptographic algorithms.* This optional section describes the Data Encryption Standard (DES) and the Rivest, Shamir, and Adleman (RSA) encryption algorithm.

11.1 SECURITY AND CRYPTOGRAPHIC ALGORITHMS

Public packet-switching networks such as the Internet traditionally have not been secure in the sense of providing high levels of security for the information that is transmitted. As

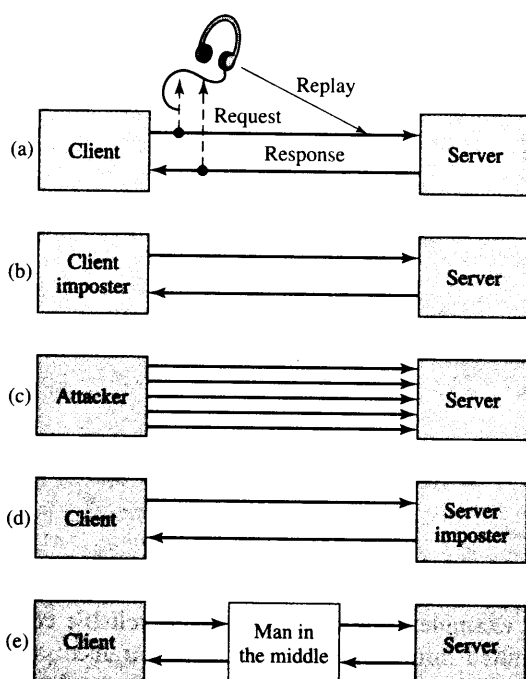


FIGURE 11.1 Network security threats: (a) eavesdropping, (b) client imposter, (c) denial of service, (d) server imposter, (e) man-in-the-middle.

these networks are increasingly used for commercial transactions, the need to provide security becomes critical. We showed earlier that many transactions take the form of a client/server interaction. Figure 11.1 shows several threats that can arise in a network setting:

- Information transmitted over the network is not secure and can be observed and recorded by eavesdroppers. This information can be replayed in attempts to access the server.
- Imposters can attempt to gain unauthorized access to a server, for example, a bank account or a database of personal records.
- An attacker can also flood a server with requests, overloading the server resources and resulting in a *denial of service* to legitimate clients.
- An imposter can impersonate a legitimate server and gain sensitive information from a client, for example, a bank account number and associated user password.
- An imposter manages to place itself as the *man in the middle*, convincing the server that it is the legitimate client and the legitimate client that it is the legitimate server.

These threats give rise to one or more of the following security requirements for information that is transmitted over a network:

- **Privacy or confidentiality:** The information should be readable only by the intended recipient.
- **Integrity:** The recipient of the information should be able to confirm that a message has not been altered during transmission.

- **Authentication:** It should be possible to verify that the sender or receiver is who he or she claims to be.
- **Nonrepudiation:** The sender cannot deny having sent a given message.¹

These requirements are not new; they take place whenever people interact. Consequently, people have developed various means to provide these security needs. For example, the privacy and integrity of important and sensitive information is maintained by keeping it under lock and key. In sensitive situations people are required to identify themselves with official documentation. Contracts are signed so that the parties cannot repudiate an agreement. In this and subsequent sections we are interested in developing analogous techniques that can be used to provide security across a network.

The need for security in communications is in fact also not new. This need has existed in military communications for thousands of years. It should not be surprising then that the approaches developed by the military form the basis for providing security in modern networks. In particular, the possession of secrets, such as passwords and keys, and the use of encryption form the basis for network security.

One feature that is new in the threats faced in computer networks is the *speed* with which break-in attempts can be made from a *distance* by multiple coordinated attackers using a network. Because the threats are implemented on computers, very high attempt rates are possible. Countermeasures include a wide range of elements from firewalls and network security protocols to security practices within an organization. In this chapter we focus on network protocols that provide security services. We emphasize that the overall security of a system depends on many factors, only some of which are addressed by network protocols.

In the following section we present an overview of basic cryptographic techniques and show how they can be used to meet the security requirements. We then discuss security in the context of peer-to-peer protocols. Finally, we show how these protocols are used in Internet security standards.

11.1.1 Applications of Cryptography to Security

The science and art of manipulating messages to make them secure is called **cryptography**. An original message to be transformed is called the **plaintext**, and the resulting message after the transformation is called the **ciphertext**. The process of converting the plaintext into ciphertext is called **encryption**. The reverse process is called **decryption**. The algorithm used for encryption and decryption is often called a **cipher**. Typically, encryption and decryption require the use of a *secret key*. The objective is to design an encryption technique so that it would be very difficult if not impossible for an unauthorized party to understand the contents of the ciphertext. A user can recover the original message only by decrypting the ciphertext using the secret key.

For example, **substitution ciphers** are a common technique for altering messages in games and puzzles. Each letter of the alphabet is mapped into another letter. The

¹On the other hand, some situations require *repudiation*, which enables a participant to plausibly deny that it was involved in a given exchange.

ciphertext is obtained by applying the substitution defined by the mapping to the plaintext. For example, consider the following substitution:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
z y x w v u t s r q p o n m l k j i h g f e d c b a
```

where each letter in the first row is mapped into the corresponding letter in the row below. The message “hvxfirgb” is easy to decode if you know the key, which is the permutation that is applied.

Transposition ciphers are another type of encryption scheme. Here the order in which the letters of the message appear is altered. For example, the letters may be written into an array in one order and read out in a different order. If the receiver knows the appropriate manner in which the reading and writing is done, then it can decipher the message. Substitution and transposition techniques are easily broken. For example, if enough ciphertext is available, such as through eavesdropping, then the frequency of letters, pairs of letters, and so forth can be used to identify the encryption scheme.

Modern encryption algorithms depend on the use of mathematical problems that are easy to solve when a key is known and that become extremely difficult to solve without the key. There are several types of encryption algorithms, and we discuss these next.

SECRET KEY CRYPTOGRAPHY

Figure 11.2 depicts a **secret key cryptographic** system where a sender converts the plaintext P into ciphertext $C = E_K(P)$ before transmitting the original message over an insecure channel. The sender uses a **secret key** K for the encryption. When the receiver receives the ciphertext C , the receiver recovers the plaintext by performing decryption $D_K(C)$, using the same key K . It is the sharing of a secret, that is, the key, that enables the transmitter and receiver to communicate. Symbolically, we can write $P = D_K(E_K(P))$. Secret key cryptography is also referred to as *symmetric key cryptography*.

The selection of the cryptographic method must meet several requirements. First of all, the method should be easy to implement, and it should be deployable on large scale. This suggests adopting a standard mathematical algorithm that can be readily implemented. On the other hand, the algorithm must provide security to all of its users. It is here that the use of a key plays an essential role. The key must uniquely specify a particular variation of the algorithm that will produce secure ciphertext. Indeed the best algorithms should prevent an attacker from deriving the key even when a large sample of the plaintext and corresponding ciphertext is known. In general, the number of possible keys must be very large. Otherwise, a brute force approach of trying all possible keys may be used to break the secret key.

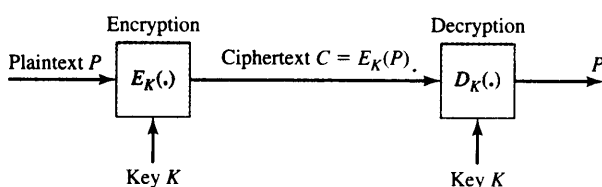


FIGURE 11.2 Secret key cryptography.

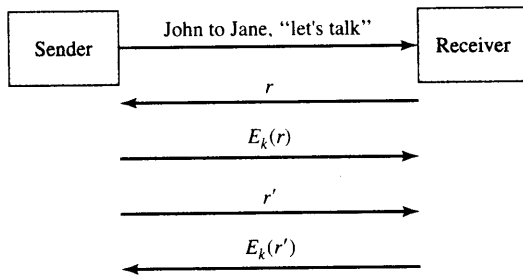


FIGURE 11.3 Secret key authentication.

Clearly, secret key cryptography addresses the *privacy* requirement. A message that needs to be kept confidential is encrypted prior to transmission, and any eavesdropper that manages to gain access to the ciphertext will be unable to access the contents of the plaintext message. The Data Encryption Standard (DES) is a well-known example of a secret key system and is discussed in a later section.

A traditional method of *authentication* involves demonstrating possession of a secret. For example, in a military setting a messenger might be confirmed to be authentic if he or she can produce the correct answer to the specific question. A similar procedure can be used over a network, using secret key cryptography. Suppose that a transmitter wants to communicate with a receiver as shown in Figure 11.3 and that the receiver and transmitter share a secret key. The transmitter sends a message identifying itself. The receiver replies with a message that contains a random number r . This process is called a **challenge**. The transmitter then sends a **response** with an encrypted version of the random number. The receiver applies the shared key to decrypt the number. If the decrypted number is r , then the receiver knows that it is communicating with the given transmitter. If the transmitter also wishes to authenticate the receiver, the transmitter can then issue a challenge by sending its own random number. It is extremely important that the random numbers used in each challenge be different; otherwise, an eavesdropper, such as the one in Figure 11.1, could maintain a table of challenges and responses and eventually manage to be authenticated. The term **nonce**, derived from number *once*, describes the desired random numbers.

Integrity refers to the ability to ascertain that a message has not been altered during transmission. One approach is to encrypt the entire message, since the modified ciphertext, when decrypted, will produce gibberish. However, in general the message itself may not have been encrypted prior to transmission. The next section discusses how integrity can be provided in this case.

CRYPTOGRAPHIC CHECKSUMS AND HASHES

The usual approach to providing integrity is to transmit a *cryptographic checksum* or *hash* along with the unencrypted message. The transmitter and receiver share a secret key that allows them to calculate the checksum that consists of a fixed number of bits. To ascertain integrity, the receiver calculates the checksum of the received message and compares it to the received checksum. If the checksums agree, the message is accepted. Note that this procedure corresponds exactly to that used in error detection in Chapter 3.

A cryptographic checksum must be designed so that it is *one way* in that it is extremely difficult to find a message that produced a given checksum. Furthermore, given a message, finding another message that would produce the same checksum should also be extremely difficult. In general the checksum is much shorter than the transmitted message. However, the cryptographic checksum cannot be too short. For example, suppose that we are dealing with messages that are 1000 bits long and that we are using a checksum that is 128 bits long. There are 2^{1000} possible messages and “only” 2^{128} possible checksums, so on average $2^{1000}/2^{128} = 2^{872}$ messages produce the same checksum. This calculation does not mean that it is easy to find a message that corresponds to a given checksum. Suppose we start going through all 2^{1000} possible messages, computing their checksum, and comparing to the desired checksum. The fraction of messages with the desired checksum is $1/2^{128}$, so on average it will take us 2^{128} tries to get to the first message that matches the desired checksum. If it takes 1 microsecond to generate and check a message, then the average time required to find one that matches the checksum is 10^{25} years.

The **message digest 5 (MD5) algorithm** is an example of a hash algorithm. The MD5 algorithm begins by taking a message of arbitrary length and padding it into a multiple of 512 bits. A buffer of 128 bits is then initialized to a given value. At each step the algorithm modifies the content of the buffer according to the next 512-bit block. When the process is completed, the buffer holds the 128-bit “hash” code. The MD5 algorithm itself does not require a key.

The **keyed MD5**, which combines a secret key with the MD5 algorithm, is widely used to produce a cryptographic checksum. First the message is padded to a multiple of 512 bits. The secret key is also padded to 512 bits and attached to the front and back of the padded message. The MD5 algorithm then computes the hash code [Kaufman et al., 1995, p. 120]. Note that in addition to the key, an ID and other information could be appended. This technique would also allow the receiver to authenticate that the authorized sender sent the information. A hash function that depends on a secret key and on a message is called a **message authentication code**.

The **secure hash algorithm 1 (SHA-1)** is another example of a hash function. SHA-1 was developed for use with the Digital Signature Standard. SHA-1 produces an 160-bit hash and is considered more secure than MD5. A keyed SHA-1 hash is produced in the same manner as a keyed MD5 hash.

A general method for improving the strength of a given hash function is to use the **hashed message authentication code (HMAC)** method. Using MD5 as an example, HMAC works as follows. First, the shared secret is padded with zeros to 512 bits. The result is XORed with *ipad*, which consists of 64 repetitions of 00110110. Second, the message is padded to a multiple of 512 bits. Third, the concatenation of the blocks in the first two steps is applied to the MD5 algorithm to obtain a 128-bit hash. The hash is padded to 512 bits. Fourth, the shared secret is padded with zeros to 512 bits, and the result is XORed with *opad*, which consists of 64 repetitions of 01011010. Fifth, the blocks in the previous two steps are applied to the MD5 algorithm to produce the final 128-bit hash. The general HMAC procedure involves adjusting the block size (512 bits for MD5) and the hash size (128 bits for MD5) to the particular hash function. For example, SHA-1 works with a block size of 512 and a hash size of 160 bits.

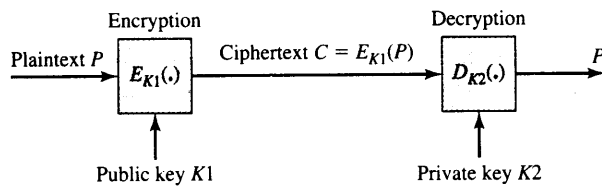


FIGURE 11.4 Public key cryptography.

PUBLIC KEY CRYPTOGRAPHY

Unlike secret key cryptography, keys are not shared between senders and receivers in **public key cryptography** (sometimes also referred to as *asymmetric cryptography*). Public key cryptography was invented in 1975 by Diffie and Hellman. It relies on two different keys, a public key and a private key. A sender encrypts the plaintext by using a **public key**, and a receiver decrypts the ciphertext by using a private key, as illustrated in Figure 11.4. Symbolically, a public key cryptographic system can be expressed as $P = D_{K2}(E_{K1}(P))$, where $K1$ is the public key and $K2$ is the **private key**. In some systems the encryption and decryption process can be applied in the reverse order such as $P = E_{K1}(D_{K2}(P))$. One important requirement for public key cryptography is that it must not be possible to determine $K2$ from $K1$. In general the public key is small, and the private key is large. The best-known example of public key cryptography is the one developed by Rivest, Shamir, and Adleman, known as **RSA**.²

Public key cryptography provides for *privacy* as follows. The transmitter uses the public key $K1$ to encrypt its message P and then transmits the corresponding ciphertext $E_{K1}(P)$ to the receiver. Only the holder of the private key $K2$ can decrypt the message $P = D_{K2}(E_{K1}(P))$; therefore, the privacy of the message is assured. Similarly, the messages from the receiver to the transmitter are kept private by encrypting them with the transmitter's public key. Note that *integrity* is not assured, since an intruder can intercept the message from the transmitter and insert a new message using the public key $K1$. This problem can be addressed by having the transmitter encrypt the message with its *private* key $K2'$ and transmit $E_{K1}(P')$, where $P' = D_{K2'}(P)$. No intruder can successfully alter this ciphertext, and the receiver can decrypt it by applying D_{K2} to $E_{K1}(P')$ to recover P' , followed by $E_{K1'}(P') = E_{K1'}(D_{K2'}(P))$ to recover P .

Public key cryptography can also be used for *authentication* as shown in Figure 11.5. Here the transmitter begins by identifying itself. The receiver picks a nonce r , encrypts it by using the transmitter's public key, and issues a challenge. The transmitter uses its private key to determine the nonce and responds with the nonce r .

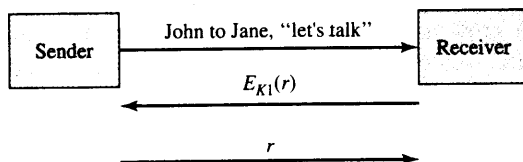


FIGURE 11.5 Public key authentication.

²RSA is described in Section 11.3.2.

Public key cryptography can also be used to provide *nonrepudiation* by producing a **digital signature**. To sign a message the transmitter first produces a noncryptographic checksum or hash of the message. The transmitter then encrypts the checksum or hash using its private key to produce the signature. No one else can create such a signature. The transmitter then sends the message and the signature to the receiver. The receiver confirms the signature as follows. First the receiver applies the public key encryption algorithm to the signature to obtain a checksum. The receiver then computes the checksum directly from the message. If the two checksums agree, then only the given transmitter could have issued the message. Note that the digital signature confirms that the transmitter produced the message and that the message has not been altered.

COMPARISON OF SECRET KEY AND PUBLIC KEY CRYPTOGRAPHIC SYSTEMS

In terms of capabilities public key systems are more powerful than secret key systems. In addition to providing for integrity, authentication, and privacy, public key systems also provide for digital signatures. Unfortunately, public key cryptography has a big drawback in that it is much slower than secret key cryptography. For this reason, public key cryptography is usually used only during the setup of a session to establish a so-called *session key*. The session key is then used in a secret key system for encrypting messages for the duration of the session.

EXAMPLE Pretty Good Privacy

Pretty Good Privacy (PGP) is a secure e-mail protocol developed by Phillip Zimmerman [Zimmerman 1995]. PGP became notorious for being freely available in the Internet in 1991 in violation of U.S. government export restrictions. The PGP software has become the defacto standard for e-mail encryption. The protocol uses public key cryptography and so it provides privacy, integrity, authentication, and digital signatures. PGP can also be used to provide privacy and integrity for stored files.

11.1.2 Key Distribution

In principle, secret key systems require every pair of users to share a separate key. Consequently, the number of keys can grow as the square of the number of users, making these systems unfeasible for large-scale use. This problem can be addressed through the introduction of a **key distribution center (KDC)** as shown in Figure 11.6. Every user has a shared secret key with the KDC. If user A wants to communicate with user B, user A contacts the KDC to request a key for use with user B. The KDC authenticates user A, selects a key K_{AB} , and encrypts it by using its shared keys with A and B to produce $E_{KA}(K_{AB})$ and $E_{KB}(K_{AB})$. The KDC sends both versions of the encrypted key to A. Finally, user A can contact user B and provide a *ticket* in the form of $E_{KB}(K_{AB})$ that allows them to communicate securely.

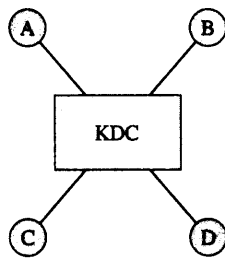


FIGURE 11.6 Key distribution.

EXAMPLE Kerberos Authentication Service

Kerberos is an authentication service designed to allow clients to access servers in a secure manner over a network. Kerberos uses a KDC that shares a secret key with every client. When a user logs on to a workstation, he supplies a name and password that is used to derive the user's secret key. The workstation is authenticated by the KDC. The KDC returns a session key encrypted with the user's secret key. The KDC also returns a ticket-granting ticket (TGT) that contains the session key and other information encrypted with the KDC's own secret key. Each time the client wishes to access a particular server, it sends a request to the KDC along with the TGT and the server's name. The KDC decrypts the TGT to recover the session key. The KDC then returns a ticket to the client that allows access to the desired server.

Public key systems require only one pair of keys per user, but they still face the problem of how the public keys are to be distributed. Because an imposter can advertise certain public keys as belonging to other parties, the public keys must be certified somehow. One approach to addressing this problem is to establish a **certification authority (CA)**. The function of a CA is to issue certificates that consist of a *signed* message, stating the name of a given user, his or her public key, a serial number identifying the certificate, and an expiration date. The certificates can be stored anywhere they can be conveniently accessed through a directory service. Each user is initially configured to have the public key of the CA. To communicate with user B, user A contacts a server to obtain a certificate for B. The signature in the certificate authenticates the message and its integrity.

X.509 AUTHENTICATION SERVICE

The ITU.500 series recommendations is intended to facilitate the interconnection of systems to provide *directory services*. A server or distributed set of servers together hold information that constitutes a *directory*. This information is used to facilitate communications between applications, people, and communication devices. An example of information provided by a directory is the mapping of user names to network addresses. The X.509 recommendation provides a framework for the provision of *authentication service* by a directory to its users. The recommendation specifies the

form of authentication information held by the directory, describes how authentication information may be obtained from the directory, states the assumptions about how authentication information is formed and placed in the directory, defines ways in which applications may use this authentication information to perform authentication, and describes how other security services may be supported by authentication. An important feature of X.509 is that the directory can hold user certificates, which can be freely communicated within the directory system and obtained by users of the directory. The certificates can contain a user's public key that has been signed by a certificate authority. Many protocols including IP Security, SSL, and TLS described later in the chapter use the X.509 certificate format.

Note that the public key of certain users will inevitably be revoked, but retrieving a certificate after it has been issued is not easy. For this reason, a *certificate revocation list (CRL)* must also be issued periodically by the CA. The CRL lists the serial numbers of certificates that are no longer valid. Thus we conclude that each time a user wishes to communicate with another, the user must retrieve not only a certificate for the given user but also a current CRL.

KEY GENERATION: DIFFIE-HELLMAN EXCHANGE

An alternative to key distribution using KDCs or CAs is to have the transmitter and receiver create a *secret shared key* by using a series of exchanges over a public network. Diffie and Hellman showed how this can be done. The procedure assumes that the transmitter and receiver have agreed on the use of a large prime number p , that is, for example, about 1000 bits long, and a *generator* number g that is less than p . The transmitter picks a random number x and calculates $T = g^x$ modulo p . Similarly the receiver picks a random number y and calculates $R = g^y$ modulo p . The transmitter sends T to the receiver, and the receiver sends R to the transmitter. At this point the transmitter and receiver both have T and R , so they can compute the following numbers:

- The transmitter calculates R^x modulo $p = (g^y)^x$ modulo $p = g^{xy}$ modulo $p = K$.
- The receiver calculates T^y modulo $p = (g^x)^y$ modulo $p = g^{xy}$ modulo $p = K$.

Therefore, both transmitter and receiver arrive at the same number K , as shown in Figure 11.7. An eavesdropper would have p , g , T , and R available, but neither x nor y . To obtain these values, the eavesdropper would need to be able to compute discrete logarithms, that is, $x = \log_g(T)$ and $y = \log_g(R)$. It turns out that this computation is exceedingly difficult to do for large numbers. Thus the transmitter and receiver jointly develop a shared secret K , which they can use in subsequent security operations.

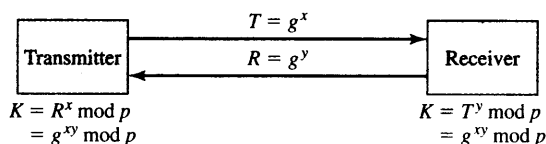


FIGURE 11.7 Diffie-Hellman exchange.

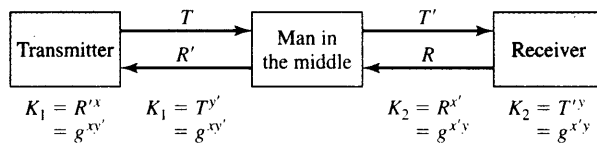


FIGURE 11.8
Man-in-the-middle attack.

The Diffie-Hellman exchange in Figure 11.7 has weaknesses. The required exponentials need many multiplications for large prime number p , so the algorithm is computationally intensive. This feature makes the system susceptible to a flooding attack, which could produce a heavy computational burden on a machine and result in denial of service to legitimate clients. In the next section we show how the use of “cookies” can thwart this type of attack.

The Diffie-Hellman exchange is also susceptible to a man-in-the-middle attack. Suppose that the intruder is able to intercept T and R as shown in Figure 11.8. The intruder keeps R and sends $R' = g^{y'}$ modulo p to the transmitter. At this point the transmitter and the intruder have established a secret key K_1 based on x and y' . Similarly, the intruder can establish a shared secret K_2 with the intended receiver based on x' and y . From now on the intruder is privy to all communications between the transmitter and receiver, and the transmitter and receiver can do nothing to detect the intruder. In the next section we show how the intruder can be prevented from inserting itself into the middle through the authentication of the transmitter and receiver and their keys T and R .

11.2 SECURITY PROTOCOLS

A security protocol refers to a set of rules governing the interaction between peer processes to provide a certain type of security service. The protocol specifies the messages that are to be exchanged, the type of processing that is to be implemented, and the actions that are to be taken when certain events occur. In this section we describe the basic protocols that have been developed for the Internet.

11.2.1 Application Scenarios

Figure 11.9 shows a number of scenarios that may require secure communication services. In part (a) two host computers communicate across the public Internet. The hosts are exposed to all the security threats that were outlined earlier, such as eavesdropping, server and client imposters, and denial-of-service attacks. The two hosts can protect some of the information transmitted by encrypting the packets that are exchanged between them. Note, however, that the routers in the Internet need to read the IP header so the header cannot be encrypted. This introduces an element of insecurity in that the source and destination addresses reveal the existence of the hosts, their addresses, and the fact that they are communicating with each other. A patient eavesdropper can also analyze the timing and frequency of packet exchanges and the length of the messages to gain some insights into the nature of the interaction. Note that the hosts in

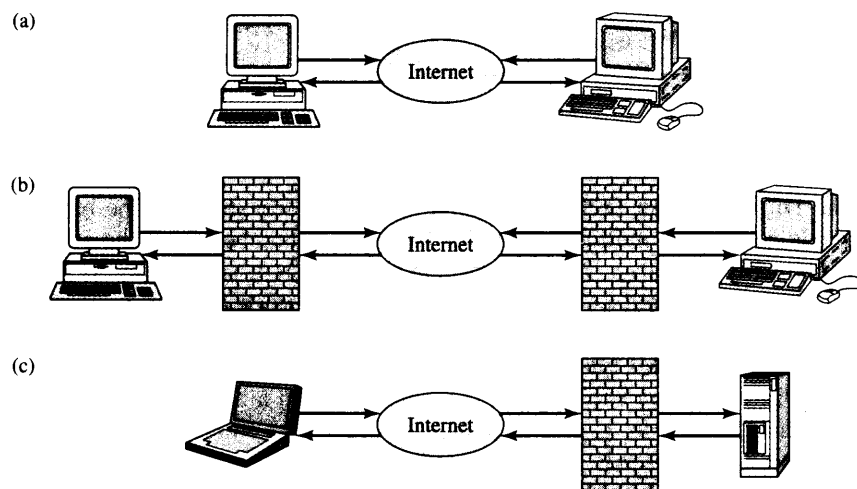


FIGURE 11.9 Scenarios requiring secure communication services.

Figure 11.9a can choose which layer to apply security to. Instead of encrypting the payloads of IP packets, the hosts might encrypt the payloads of TCP segments, or only certain application layer PDUs.

In part (b) two host computers on local networks have gateways between their local networks and the public Internet. These gateways could be **firewalls** whose role is to enforce a security policy between the secure internal networks and the Internet. A firewall is implemented in a computer or a router, and its role is to control external access to internal information and services. A simple firewall system may involve *packet filtering* carried out by a router to enforce certain rules. Various fields in arriving packets are examined to determine whether packets should be allowed to pass or be discarded. These fields can include source and destination IP addresses and TCP/UDP port numbers, ICMP message types, and fields inside the IP and TCP payloads. Secure communications between host A and host B in Figure 11.9b can be provided by establishing a secure *tunnel* between the two gateways. The packets that are exchanged between the hosts can then be completely encrypted, so the internal addresses of the packets are secure as well. Part (c) shows a situation that is typical when a mobile host attempts to access its home office across the public Internet. The mobile host and the home gateway must have procedures in place to allow the mobile host access while barring intruders.

Firewall systems can include elements that operate at layers above the network layer. For instance, packet filtering cannot understand the contents, for example, application-level commands, in the payload. For this reason, *application-level gateways* or *proxies* have been developed to enforce security policy at the level of specific services. An application-level gateway is interposed between the actual client and the actual server: The gateway monitors and filters the messages and relays them from client to server and from server to client. The gateway can be configured so that users need to provide authentication information and so that only certain features of the service are made available. Application-level gateways for Telnet, FTP, and HTTP are in common use.

The *circuit-level gateway* is another type of firewall system. This gateway restricts the TCP connections that are allowed across it. An end-to-end connection between a client and a server is not allowed and instead must be broken into a connection from the client to the gateway and a connection from the gateway to the server. Once a user has been authenticated, the gateway relays segments between the two end systems.

11.2.2 Types of Service

In this section we are concerned with three communication security requirements:

1. *Integrity*: The recipient of the information from the network should be able to confirm that the message has not been altered during transmission.
2. *Authentication*: The recipient should be able to ascertain that this sender is who he or she claims to be.
3. *Privacy*: The information should be readable only by the intended recipient.

In network applications that require security, the first two requirements are usually essential. The third requirement involves encryption of information that can entail significant additional processing and hence is not always justified.

First we consider the case of two peer processes that have already established a **security association**, which specifies the type of processing to be carried out by the processes, including the type of cryptographic algorithms and the shared secret keys. Let us consider how a communication service can provide integrity, authentication, and privacy.

INTEGRITY AND AUTHENTICATION SERVICE

We saw in Section 11.1 that integrity can be provided through the use of a *cryptographic checksum* algorithm that takes a concatenation of the shared secret key and the message and produces a shorter fixed-length message using a one-way function. The algorithm for producing the checksum or hash is public, but without the secret key it is extremely difficult to modify the message and still produce the right checksum. On the other hand, the receiver can easily verify that the received information has not been altered. It simply concatenates the shared secret key with the received message to recalculate the checksum. If the recalculated and the received checksums are the same, then the receiver can be very confident that the message was not altered. If they do not agree, then the receiver can be certain that something is wrong. Note that the checksum calculation can be extended to cover any information whose integrity needs to be verified. In particular, if the cryptographic checksum includes the shared secret key, the sender's identity, and the message, then the identity of the sender is also authenticated.

Figure 11.10a shows a typical packet structure for providing integrity and authentication service. An **authentication header** is sandwiched in between the normal packet header and its payload (SDU). The normal header includes a field set to indicate the presence of the authentication header. The authentication header itself contains a field that identifies the security association, a field for a sequence number, and a field for the cryptographic checksum. Ideally, the cryptographic checksum should cover the entire

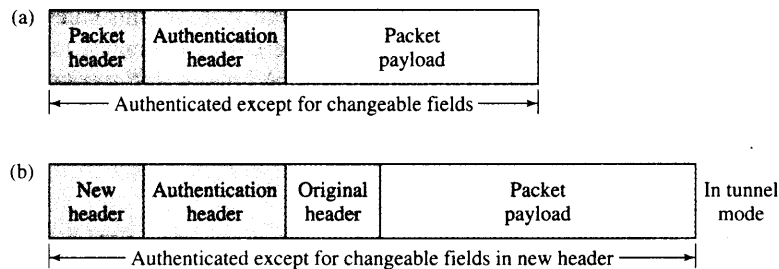


FIGURE 11.10 Packet structure for authentication and integrity service.

packet. However, some fields in the packet header need to be changed while the packet traverses the network, and so these need to be excluded. The cryptographic checksum algorithm is applied to the packet in Figure 11.10a with the values in the changeable fields and the cryptographic checksum field set to zero. The resulting checksum is then inserted in the authentication header, and the packet is transmitted.

To verify integrity and authentication, the receiver recalculates the cryptographic checksum based on the received packet and the shared secret key, with the appropriate fields set to zero. If the recalculated checksum and the received checksum do not agree, the packet is discarded and the event is recorded in an audit log.

The authentication header can have a *sequence number* whose purpose is to provide protection against **replay attacks**. A typical ploy by eavesdroppers is to replay previously recorded sequences of packets to gain access to a system. When a security association is established, the sequence number is set to zero. Each time a new packet is sent, it receives a new sequence number. The transmitter has the task of ensuring that a sequence number is never reused. Instead when the number space is exhausted, the security association will be closed and a new one will be established. A receiver is prepared to accept a packet only *once*, so a replay attack will not work. Because packets can arrive out of order, at any given time the receiver maintains a sliding window of sequence numbers. Packets with sequence numbers to the left of the window are rejected. Packets inside the window are checked against a list to see whether they are new, and if so, their cryptographic checksum is checked. Packets to the right of the window are treated as new, and their cryptographic checksum is checked.

A **tunnel** can be established to provide security between two gateways only as is shown in Figure 11.11. The tunnel is established by encapsulating a packet inside

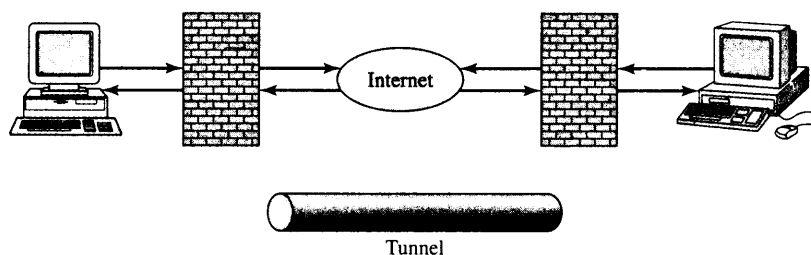


FIGURE 11.11 Tunnel between two firewall systems.

another packet as shown in Figure 11.10b. Here the header in the inner packet has the address of the destination host, and the header in the outer packet has the address of the gateway. In tunnel mode the authentication header can protect the *entire* inner packet and the unchangeable part of the outer packet.

PRIVACY SERVICE

The integrity and authentication service do not prevent eavesdroppers from reading the information in each packet. Encryption is required to provide privacy in the transmission of packet information. Again we suppose that a security association is already in place, so the transmitter and receiver have already agreed on the cryptographic algorithms and shared secret keys that are to be used.

Figure 11.12a shows a typical packet structure where an **encryption header** is inserted after the normal header. The normal header contains a field set to indicate the presence of an encryption header. The encryption header contains fields to identify the security association and to provide a sequence number. This header is followed by an encrypted version of the payload and possibly by initialization, padding, and other control information. Note that the packet header and the encryption header are *not* encrypted, so privacy is provided only by the upper-layer protocols encapsulated in the payload. The receiver takes each packet and decrypts the payload. If the payload portion has been altered during transmission, then the decrypted message will differ from the original message. Such changes may be detectable by the higher-layer protocol if the decrypted message does not make sense.

Figure 11.12b shows a packet structure that can be used to provide privacy for the payload and integrity and authentication for the overall packet. The packet header contains a field indicating that an authentication header follows. The authentication header in turn contains a field indicating that an encryption header follows. The payload is encrypted first, and then a cryptographic checksum over the entire packet is calculated with some of the fields set to zero as before. This packet structure also provides the antireplay capability.

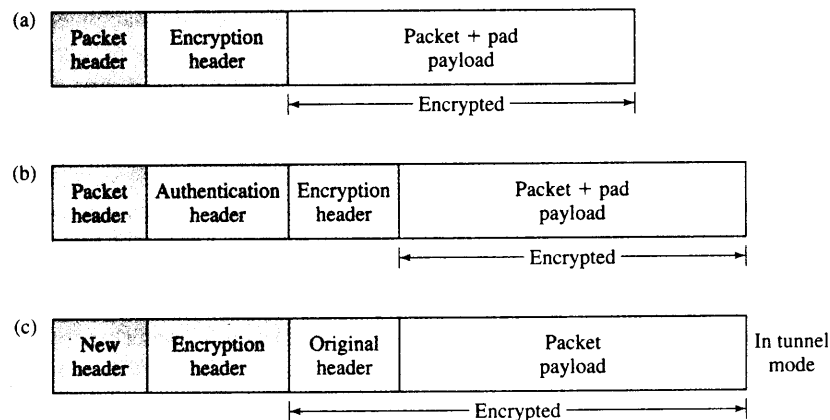


FIGURE 11.12 Packet structure for privacy service.

These packet structures allow eavesdroppers to read the information in the packet header. A stronger form of privacy is obtained by using encryption in a *tunnel mode* as shown in Figure 11.12c. The inner packet header carries the addresses of the ultimate source and destination, and the outer packet header may contain other addresses, for example, those of security gateways. All information after the encryption header is encrypted and hence not readable by an eavesdropper.

There are many options for combining the application of authentication and encryption headers in normal and tunnel modes. We explore these in the problems.

EXAMPLE Virtual Private Networks

A virtual private network (VPN) service has traditionally relied on a dedicated set of network resources (leased lines) that provide wide area connectivity for different sites of a large company. The low cost of Internet communications has made the establishment of VPNs across the Internet very attractive. The key requirement here is to create private communication across a public insecure medium. For this reason, VPNs over the Internet are generally based on the establishment of secure tunnels.

11.2.3 Setting up a Security Association

Suppose that two host computers wish to establish secure communications across a public network. The computers must first establish a security association. This step requires them to agree on the combination of security services they wish to use and on the type of cryptographic algorithms that are to be utilized. They also need to authenticate that the other host is who it claims to be. Finally, they must somehow establish a shared secret key that can be used in the subsequent cryptographic processing. Key distribution is a central problem in meeting these requirements and therefore enabling widespread use of security services.

One approach is to “manually” distribute keys to the appropriate machines ahead of time. This approach, however, is not very scalable and not suitable for highly dynamic environments. As discussed in Section 11.1.2, another approach is to use either key distribution centers to obtain secret keys or certification authorities that issue digitally signed certificates that provide the public keys of various users. A very appealing alternative to these approaches is a procedure that allows the two hosts to establish a security association and a common secret key *independently* of other hosts or servers. In this section we describe the **Internet key exchange (IKE)** protocol that has been developed to provide such a procedure.

We saw that the Diffie-Hellman exchange allows two hosts to establish a common secret key through an exchange of information over an insecure network. We saw however that the exchange was susceptible to a man-in-the-middle attack where an intruder can manage to insert itself transparently between the two hosts. In addition, we saw that the exchange made the hosts susceptible to denial-of-service attacks. The IKE protocol addresses these problems by adding an authentication step after the

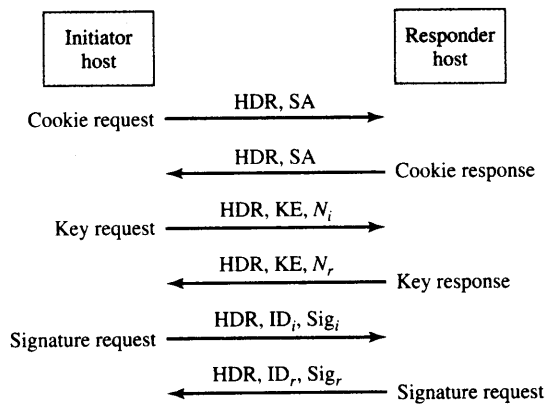


FIGURE 11.13 Establishing a security association.

Diffie-Hellman exchange that can detect the presence of a man-in-the-middle and by using “cookies” to thwart denial-of-service attacks.

Figure 11.13 shows a series of message exchanges between two hosts to establish a security association. The exchange makes use of a digital signature for authentication and features the use of a pair of cookies generated by the hosts to identify the security association and to prevent flooding attacks. The cookie generation must be fast and must depend on the source and destination address, the date and time, and a local secret. A cryptographic hash algorithm such as MD5 can be used for this purpose. To protect themselves against flooding attacks, the hosts always check for the validity of the cookies in arriving packets first.

Note the following about the exchange of messages:

- The initiator first generates a unique pseudorandom number of, say, 64 bits, which is called the initiator’s cookie C_i . The initiator associates this cookie value with the expected address of the responder. The initiator host sends a *cookie request* message to the responder host requesting a security association. The header (HDR) contains the initiator’s cookie. The *security association (SA) field* in the message offers a set of choices regarding encryption algorithm, hash algorithm, authentication method, and information about the parameters to be used in the Diffie-Hellman exchange.
- The responder checks to see whether the initiator’s cookie is not already in use by the source address in the packet header. If not, the responder generates its cookie C_r . The responder associates this cookie value with the expected address of the initiator. The responder replies with a *cookie response* message in which it selects one of the offered choices in the initiator’s SA field. The header includes both cookies, C_r and C_i .
- Upon receiving the response, the initiator first checks the address and initiator cookie in the arriving packet against its list. From now on the initiator will identify the security association by the pair (C_i, C_r) . At this point it records the association as “unauthenticated.” Next the initiator sends a *key request* message including its public Diffie-Hellman value $T = g^x$ modulo p and a nonce N_i .
- The responder host first checks the responder cookie in the arriving message. If the cookie is not valid, the message is ignored. If the cookie is valid, the security association will henceforth be identified by the pair (C_i, C_r) . At this point the association

is recorded as “unauthenticated.” The responder sends a *key response* message with its public value $R = g^y$ modulo p and a nonce N_r .

- After this exchange, both the initiator and responder hosts have the secret constant $K = g^{xy}$ modulo p . Both parties now compute a secret string of bits SKEYID known only to them; for example, they might obtain a hash of the concatenation of the two nonces N_i and N_r and K . SKEYID might be 128 bits long.
- The initiator now prepares a signature stating what it knows: namely, SKEYID, T , R , C_i , C_r ; the contents of the SA field; and the initiator identification. For example, the initiator can obtain a hash of the concatenation of the binary representations of all this information. The initiator identification and the signature are then encrypted with an algorithm specified in the security association by using a key derived from K , and (C_i, C_r) ; which are known to the initiator and responder. The initiator sends this information in a *signature request* message.
- The responder decrypts the message from the initiator. The responder then recalculates the hash of its version of the shared information, namely, SKEYID, T , R , C_i , C_r ; the contents of the SA field; and the initiator identification. If the recalculated hash agrees with the received hash, then the initiator and the Diffie-Hellman public values have been authenticated. The security association and keys are recorded as authenticated. A man in the middle would have been detected at this point, since it could not have knowledge of SKEYID, which was derived from K .
- The responder now prepares its signature stating what it knows: namely, SKEYID, R , T , C_i , C_r ; the contents of the SA field; and the responder identification. The responder identification and the signature are then encrypted, and the *signature response* message is sent to the initiator.
- The initiator recalculates the hash of its version of the shared information to authenticate the responder as well as the values of the Diffie-Hellman public values. At this point the security association is established. The security association and keys are recorded as authenticated.

Once the security association is established, the two hosts can derive additional shared secret keys from the cookie values and K . The availability of shared secret key information also allows the hosts to quickly establish additional security associations as needed.

11.2.4 IPSec

The goal of **IP Security (IPSec)** is to provide a set of facilities that support security services such as authentication, integrity, confidentiality, and access control at the IP layer. IPSec also provides a key management protocol to enable automatic key distribution techniques. The security service can be provided between a pair of communication nodes, where the node can be a host or a gateway (router or firewall).

IPSec uses two protocols to provide traffic security: *authentication header* and *encapsulating security payload*. These protocols may be applied alone or together to provide a specific security service. Each protocol can operate in either **transport mode** or **tunnel mode**. In the transport mode the protocols provide security service to

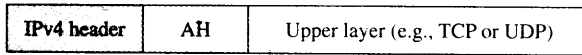


FIGURE 11.14 Example with IPv4.

the upper-layer protocols. In the tunnel mode the protocols provide security service to the tunneled packets.

AUTHENTICATION HEADER

Authentication and integrity of an IP packet can be provided by an **authentication header (AH)**. The location of the AH is after the headers that are examined at each hop and before any other headers that are not examined at an intermediate hop. With IPv4, the AH immediately follows the IPv4 header, as shown in Figure 11.14.³ The protocol value in the IP header is set to 51 to identify the presence of an AH following the IP header. The authentication information is calculated over the entire IP packet, including the IP header, except over those fields that change in transit (i.e., mutable fields such as TTL, header checksum, and fragment offset) that are set to zero in the calculation.

The format of the AH is shown in Figure 11.15. The next header field is used to identify the next payload after the AH. In other words, it has the same purpose as the IP protocol number in IPv4. The length field indicates the length of the authentication data in multiples of four octets. The security parameters index (SPI) identifies the security association for this packet. The value of the sequence number field is incremented by one for each packet sent. Its purpose is to protect against replay attacks. The result of the authentication algorithm is placed into the authentication data field. AH implementations must support HMAC with MD5 and HMAC with SHA-1 for the authentication algorithms.

The AH is incorporated in IPv6. The AH appears after the hop-by-hop, routing, and fragmentation extension headers. The destination options extension header can appear either before or after the AH. The protocol header that precedes the AH contains the value 51 in the next header field.

ENCAPSULATING SECURITY PAYLOAD

The **encapsulating security payload (ESP)** provides confidentiality, authentication, and data integrity. An ESP can be applied alone or in combination with an AH.

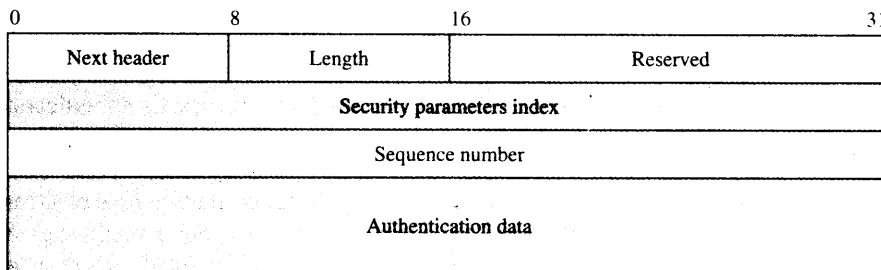


FIGURE 11.15 Format of authentication header.

³In the tunnel mode the AH is located immediately before the inner IP header.

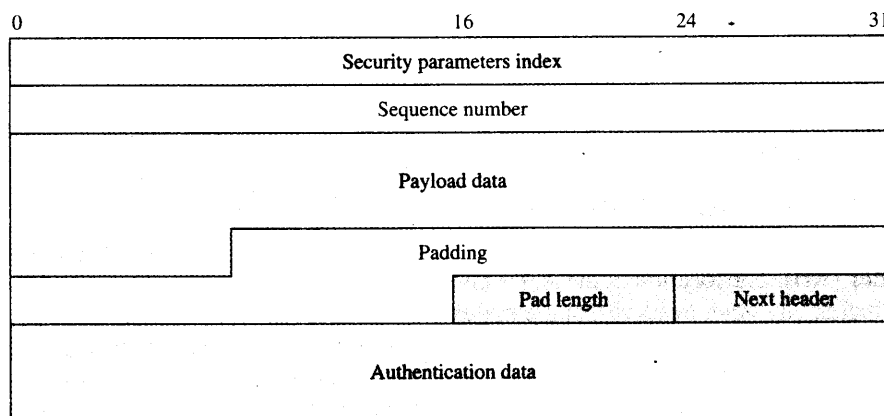


FIGURE 11.16 Format of ESP.

The ESP consists of a header and a trailer, as shown in Figure 11.16. The authenticated coverage starts from the SPI field until the next header field, and the encrypted coverage starts from the payload data field until the next header field. The protocol number immediately preceding the ESP header is 50. The SPI and sequence number have the same meanings as before. The payload data is variable-length data whose contents are specified in the next header field. The padding field is optional; its purpose is to conform to a particular encryption algorithm that requires the plaintext to be a multiple of some number of octets. The pad length field indicates the length of the padding field in octets. If the padding field is not present, then the pad length is zero. The authentication data field is optional, and its purpose is to provide authentication service.

The ESP is incorporated in IPv6 and appears after the hop-by-hop, routing, and fragmentation extension headers. The destination options extension header can appear either before or after the ESP header. The protocol header that precedes the ESP header contains the value 50 in the next header field.

11.2.5 Secure Sockets Layer and Transport Layer Security

The Secure Sockets Layer (SSL) protocol was developed by Netscape Communications to provide secure HTTP connections. SSL operates on top of a reliable stream service such as TCP and provides a secure connection for applications such as HTTP, as well as FTP, Telnet, and so on. SSL is widely used in web applications that involve electronic ordering and payments. SSL is also used in conjunction with various e-mail programs. SSL version 3.0 [Freier et al., 1996] was submitted to the IETF for standardization and, after some changes, led to the Transport Layer Security (TLS version 1.0) protocol in RFC 2246. We cover the TLS protocol as described in [RFC 2246].

As shown in the Figure 11.17, the TLS protocol consists of protocols that operate at two layers: the TLS Record protocol and the TLS Handshake protocol, along with the Change Cipher Spec protocol and the Alert protocol.

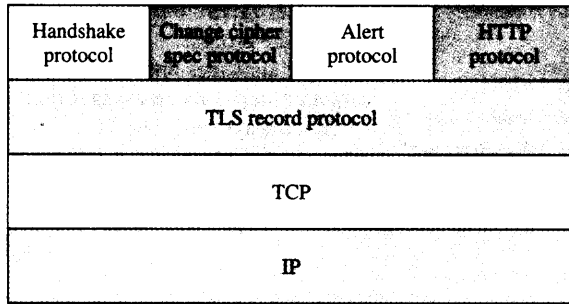


FIGURE 11.17 TLS in the TCP/IP protocol stack.

The TLS Record protocol provides a secure connection with the attributes of privacy and reliability. The connection provides *privacy* through the use of symmetric (secret key) encryption. The specific encryption algorithms can be selected from a wide range of standard algorithms. The secret keys that are used in the symmetric encryption algorithms are generated uniquely for each connection and are derived from a secret that is negotiated by another protocol, for example, the TLS Handshake protocol. The TLS Record protocol can operate without encryption. The connection provides *reliability* through the use of a keyed message authentication code (MAC).⁴ The specific hash function for a connection can be selected from a set of standard hash functions. The TLS Record protocol typically operates without a MAC only while a higher-layer protocol is negotiating the security parameters.

The TLS Handshake protocol, along with the Change Cipher Spec protocol and the Alert protocol is used to negotiate and instantiate the security parameters for the record layers, to authenticate the users, and to report error conditions. The TSL Handshake protocol is used by a server and a client to establish a **session**. The client and server negotiate parameters such as protocol version, encryption algorithm, and method for generating shared secrets. They can authenticate their identity by using asymmetric (public key) encryption with an algorithm that can be selected from a set of supported schemes. Typically only the server is authenticated to protect the user from a man-in-the-middle attack. The client and server also negotiate a shared secret that is kept secure from eavesdroppers. The negotiation is made reliable so that its parameters cannot be modified by an attacker without being detected. Once a client and server have established a session, they can set up multiple secure connections, using the parameters that have been established for the session.

THE TLS HANDSHAKE PROTOCOL

The client and server use the Handshake protocol to negotiate a session that is specified by the following parameters [RFC 2246, p. 23]:

- *Session identifier*: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

⁴Please note that MAC in this chapter refers to message authentication code, not to medium access control.

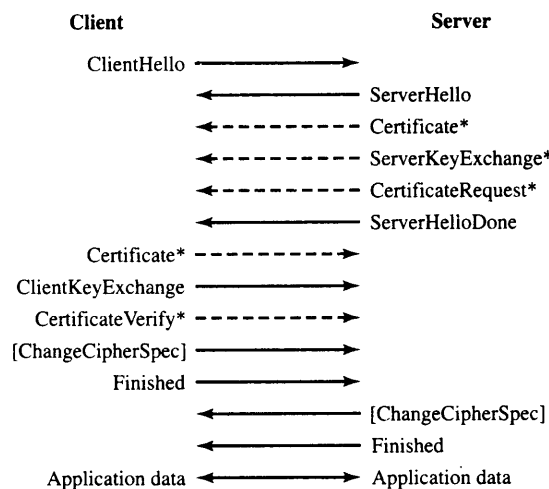


FIGURE 11.18 The TLS handshake process.

Notes: * Indicates optional or situation-dependent messages that are not always sent. The **ChangeCipherSpec** is not a TLS handshake message.

- *Peer certificate*: An X509.v3 certificate of the peer. This element of the state may be null.
- *Compression method*: The algorithm used to compress data prior to encryption.
- *Cipher spec*: Specifies the bulk data encryption algorithm (such as null or DES) and a MAC algorithm (such as MD5 or SHA). It also defines cryptographic attributes such as `hash size`.
- *Master secret*: A 48-byte secret shared between the client and the server.
- *Is resumable*: A flag indicating whether the session can be used to initiate new connections.

The TLS Record layer uses these session parameters to create its security parameters. The resumption feature allows many connections to be instantiated using the same session.

The TLS handshake process is shown in Figure 11.18.

Step 1: The client and server exchange hello messages to negotiate algorithms, exchange random values, and initiate or resume the session.

In Figure 11.18 the client sends a **ClientHello** message to request a connection. The **ClientHello** message includes the client version of the TLS protocol; the current time and date in standard UNIX 32-bit format and a 28-byte random value; an optional session ID (if not empty, this value identifies the session whose security parameters the client wishes to reuse; this field is empty if no session ID is available or if new security parameters are sought); a **CipherSuite** list that contains the combinations of key exchange, bulk encryption, and MAC algorithms that are supported by the client; and a list of compression algorithms supported by the client.

The server examines the contents of the **ClientHello** message. The server sends a **ServerHello** message if it finds an acceptable set of algorithms in the lists proposed by the client. If the server cannot find a match, it sends a handshake failure alert message and closes the connection. The **ServerHello** message contains the following parameters:

the server version, a random value that is different from and independent of the value sent by the client, a session ID, a CipherSuite selected from the list proposed by the client, and a compression algorithm from the list proposed by the client.

Step 2: The client and server exchange cryptographic parameters to allow them to agree on a premaster secret. If necessary, they exchange certificates and cryptographic information to authenticate each other. They then generate a master secret from the premaster secret and exchange random values.

In Figure 11.18, after the ServerHello message the server may also send the following three messages. The Certificate message is sent if the server needs to be authenticated. The message generally includes an X509.v3 certificate that contains a key for the corresponding key exchange method. The ServerKeyExchange is sent immediately after the Certificate message and is required when the server certificate message does not contain enough data to allow the client to exchange a premaster secret, as for example in a Diffie-Hellman exchange. The CertificateRequest message is sent by the server if the client is required to be authenticated. Finally, the server sends the ServerHelloDone message, indicating that it is done sending messages to support the key exchange. The server then waits for the client's response.

The client examines the messages from the server and prepares appropriate responses. If required to, the client sends a certificate message with a suitable certificate. If it has no suitable certificate, the client may reply with a certificate message containing no certificate, but the server may then respond with a fatal handshake failure alert message that results in a termination of the connection. The client may follow the certificate message with a ClientKeyExchange message that provides information to set the premaster secret. The CertificateVerify message is sent by a client after it has sent a certificate message. The purpose of the message is to explicitly verify the client certificate. The client prepares a digital signature of the sequence of messages that have been exchanged from the client hello up to but not including this message. The client uses its private key to prepare the signature. This step allows the server to verify that the client owns the private key for the client certificate.

Step 3. The client and server provide their record layer with the security parameters. The client and server verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

The ChangeCipherSpec message is part of the Change Cipher protocol that signals changes in the ciphering strategy. The protocol consists of a single message that is encrypted and compressed according to the current connection state. When sent by the client, the ChangeCipherSpec message notifies the server that subsequent records will be protected under a new CipherSpec and keys. After the client sends the ChangeCipherSpec message in Figure 11.18, the server copies its pending CipherSpec into the current CipherSpec. The client follows immediately with a finished message, which is prepared using the new CipherSpec algorithms. The finished message allows the server to verify that the key exchange and authentications have been successful.

The server responds with ChangeCipherSpec and finished messages of its own. Once the client and server have validated the finished messages they receive, they finally can begin to exchange information over the connection.

THE TLS RECORD PROTOCOL

The TLS Record protocol is used for the encapsulation of higher-level protocols. The protocol provides privacy through the use of symmetric encryption and provides reliability through the use of a message authentication code. The operation of the TLS Record protocol is determined by the TLS connection state, which specifies the compression, encryption, and MAC algorithms; the MAC secret; the bulk encryption keys; and the initialization vectors for the connection in the read (receive) and write (send) directions. The initial state always specifies that no encryption, compression, or MAC be used. The TLS Handshake protocol sets the security parameters for pending read and write states, and the ChangeCipherSpec protocol makes the pending states current.

The connection state also includes the following elements: the state of the compression algorithm, the current state of the encryption algorithm, the MAC secret for the connection, and the record sequence number that is initially 0 and may not exceed $2^{64} - 1$. The connection state needs to be updated each time a record is processed.

The sender in the TLS Record protocol is responsible for taking messages from the application layer, fragmenting them into manageable blocks, optionally compressing them, applying a Message Authentication Code, applying encryption, and transmitting the results. The receiver is responsible for decryption, verification, decompression, and message reassembly and delivery to the application layer.

BACKWARD COMPATIBILITY WITH SSL

The TLS v1.0 protocol is based on the SSL 3.0 but is sufficiently different to not interoperate. However, TLS 1.0 does incorporate a feature that allows a TLS implementation to back down to SSL 3.0. When a TLS client wishes to negotiate with a SSL 3.0 server, the client sends a ClientHello message using the SSL 3.0 record format and client structure but sends {3, 1} for the version field to indicate that it supports TLS 1.0. A server that supports only SSL 3.0 will respond with an SSL 3.0 ServerHello message. A server that supports TLS will respond with a TLS ServerHello. The negotiations will then proceed as appropriate.

A TLS server that wishes to handle SSL 3.0 clients should accept SSL 3.0 ClientHello messages and respond with SSL 3.0 ServerHello messages if the ClientHello message has version field {3, 0}, indicating that the client does not support TLS.

For a discussion of the differences between TLS 1.0 and SSL 3.0, see [RFC 2246].

A DETAILED EXAMPLE

Figure 11.19 and Figure 11.20 show an example packet capture sequence that transpires when a user logs in using a secure connection to a popular e-mail provider. When the user requests a secure connection, the browser program initiates a TCP connection port 443. Note in frame number 2 in the figure that https is used to denote HTTP over SSL. Once the TCP connection is set up, the client sends the ClientHello message shown in frame 5. From the middle pane in Figure 11.19, we can see that the client is

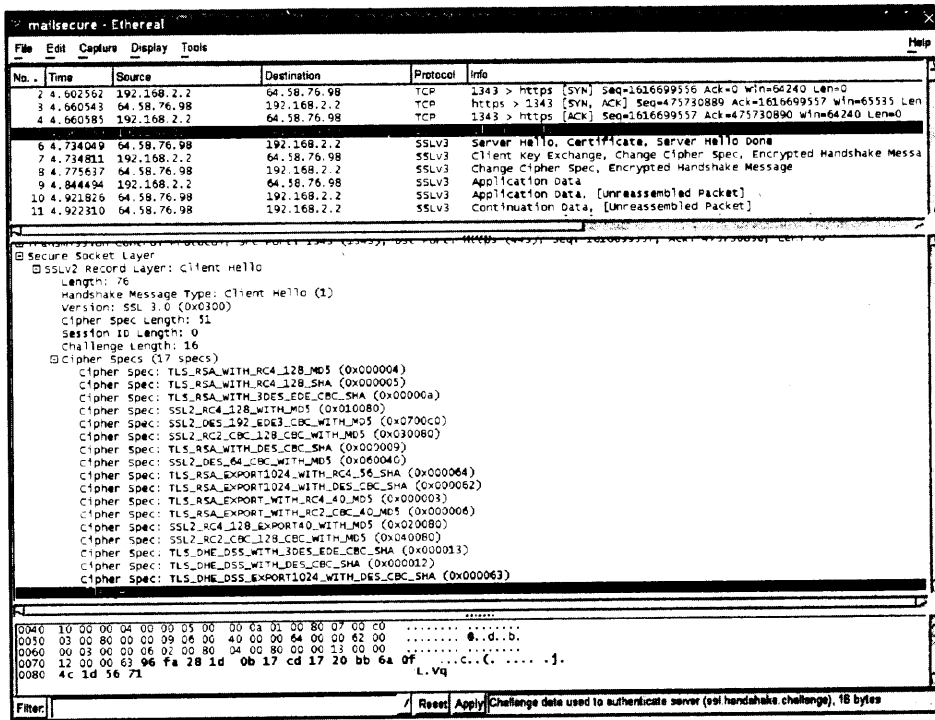


FIGURE 11.19 Example of SSL message exchange: ClientHello.

running version SSL 3.0. We can also see 17 cypher specs that are supported by the client. Finally in the third pane we see a 16-byte challenge sequence.

Figure 11.20 shows the ServerHello response from the server. From the middle pane we see that the server runs version SSL 3.0 and that it has selected the first choice in the client's cipher suite list, that is, TLS_RSA_WITH_RC4_128_MD5. This indicates that authentication will be done using RSA public key algorithm and that RC4 with a key size of 128 will be used for encryption. RC4 takes its initial key and expands it to a keystream that consists of a number of bytes that matches the number of bytes in the plaintext. The cyphertext is obtained by XORING the keystream and the plaintext. MD5 is to be used for message authentication. The server has also sent the current time and date, a sequence of 28 random bytes, which can be seen if one clicks on the random.bytes field, and a 32-byte session ID. The second part of the middle pane in frame 6 contains a certificate from the server that contains its public key. Typically a browser keeps a list of trusted certification authorities and their public keys. If the certificate received from the server is signed by a CA, the client can then authenticate the server. The third pane in Figure 11.20 contains the first part of the certificate. In the third part of the middle pane, the server sends a ServerHelloDone message.

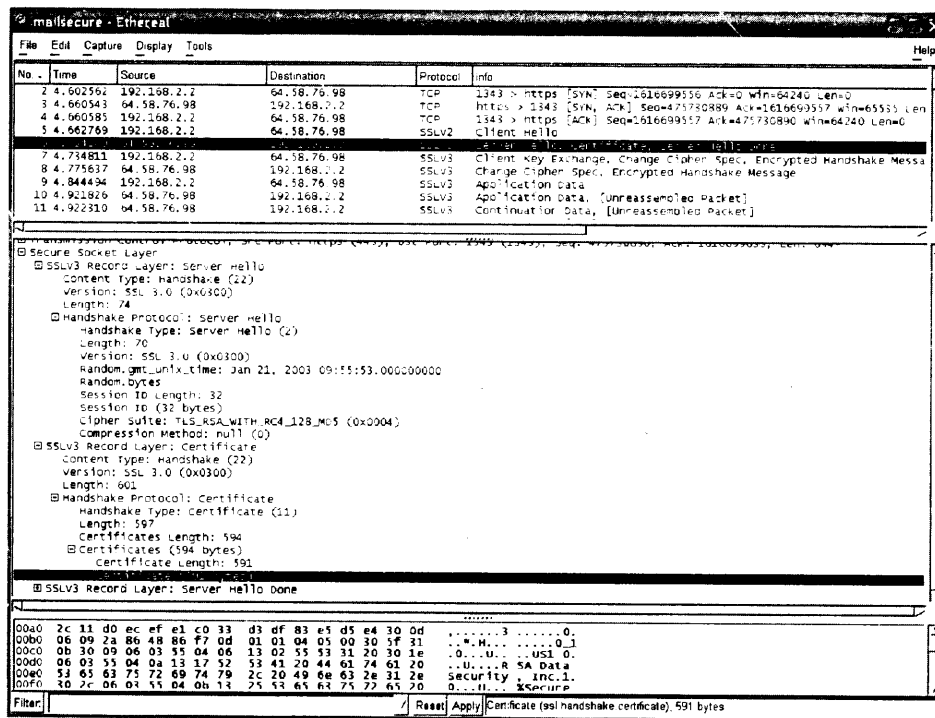


FIGURE 11.20 Example of SSL protocol exchange: ServerHello.

The client responds with frame 7, which carries ClientKeyExchange, ChangeCipherSpec, and EncryptedHandshakeMessage. Figure 11.21 shows the detail of this message. The middle pane highlights the encrypted handshake message and the bottom pane shows the corresponding 56 encrypted bytes. The server responds with frame 8 that carries ChangeCipherSpec and EncryptedHandshakeMessages. At this point the client and server have established a secure connection and can begin to exchange encrypted application messages in frame 9 and beyond.

11.2.6 802.11 and Wired Equivalent Privacy

Wireless networks provide compelling benefits to the user by enabling mobility and freedom from wires. However wireless networks pose serious challenges to security because the signals are easily captured by nearby devices. The IEEE 802.11 developed the Wired Equivalent Privacy (WEP), which operates at the data link layer to encrypt the payload in individual frames.

WEP assumes that the sender and receiver share a secret 40-bit key. A 24-bit initialization vector (IV) is generated for each frame. The secret key and the IV form a 64-bit key that is used to derive a sequence of key bytes that is the same length as the payload and an associated 32-bit CRC. The encrypted frame is obtained by taking the

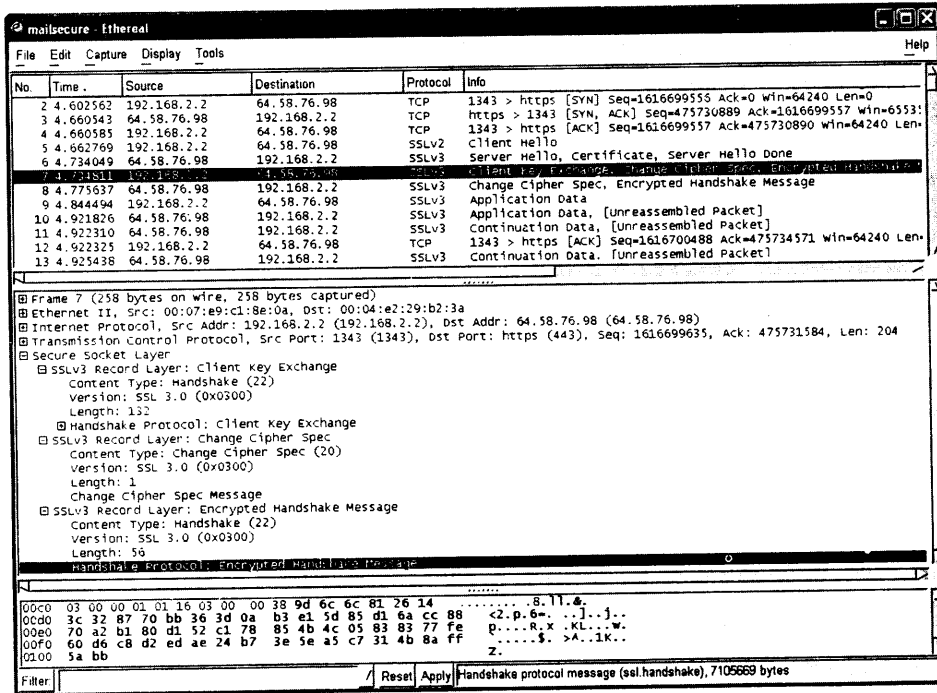


FIGURE 11.21 SSL message exchange: ClientKeyExchange, ChangeCipherSpec, EncryptedHandshakeMessage.

exclusive-OR of the sequence of key bytes and the plaintext payload and CRC. The IV is transmitted in plaintext along with encrypted frame as shown in Figure 11.22.

Fluhrer et al. developed a method for obtaining the key from observations of a sufficient number of 802.11 transmissions, and Stubbenfield et al. implemented the method and showed that the IEEE 802.11 key in WEP can be retrieved in a short period of time. The IEEE 802.11 is working on a replacement for WEP.

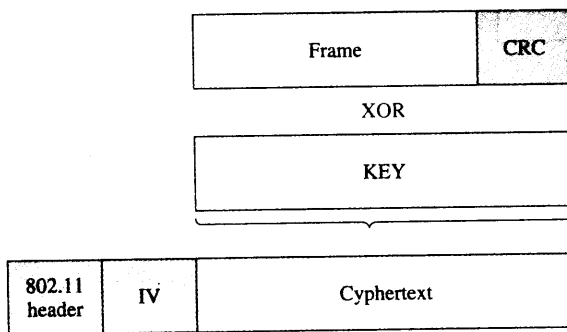


FIGURE 11.22 Wired Equivalent Privacy method.

11.3 CRYPTOGRAPHIC ALGORITHMS

We have so far intentionally avoided discussing the details of particular cryptographic algorithms. This approach enabled us to emphasize the fact that security procedures are independent of specific cryptographic algorithms as long as certain requirements are met. In this section we discuss two specific cryptographic algorithms that are used in current standards. The reader is referred to [Kaufman et al., 1995], [Schneier 1996], and [Stallings 1999] for more detailed discussions of cryptographic algorithms.

11.3.1 DES

Data Encryption Standard (DES) was developed by IBM in early 1970s and adopted by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), in 1977. DES is now the most widely used shared key cryptographic system. DES can be implemented much more efficiently in hardware than in software.

In the encryption process DES first divides the original message into blocks of 64 bits. Each block of 64-bit plaintext is separately encrypted into a block of 64-bit ciphertext. DES uses a 56-bit secret key. The choice of the key length has been a controversial subject. It is generally agreed that 56 bits are too small to be secure. The DES encryption algorithm, which has 19 steps, is outlined in Figure 11.23. The decryption basically runs the algorithm in reverse order.

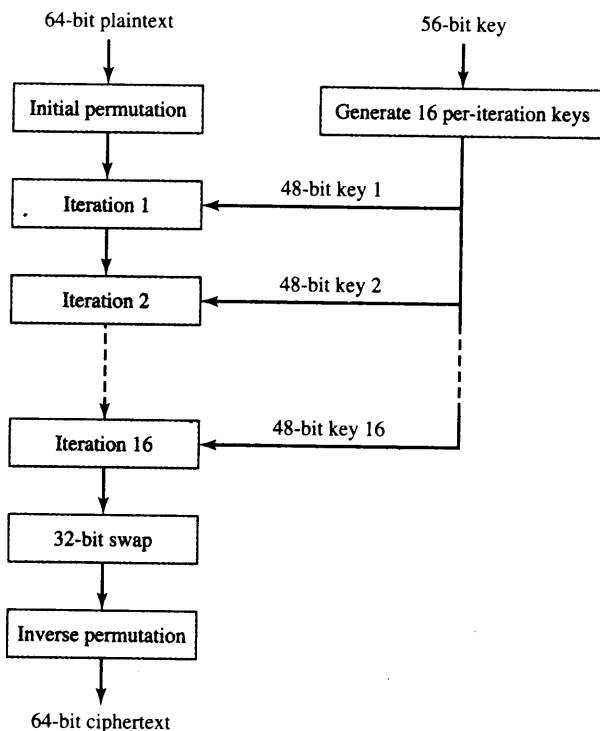


FIGURE 11.23 Outline of DES algorithm.

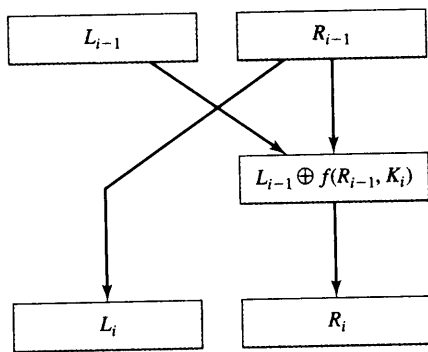


FIGURE 11.24 Each iteration in DES.

Each step in the DES algorithm takes a 64-bit input from the preceding step and produces a 64-bit output for the next step. The first step performs an initial permutation of 64-bit plaintext that is independent of the key. The last step performs a final permutation that is the inverse of the initial permutation. It is generally believed that the initial and final permutations do not make the algorithm more secure. However, they may add some value when multiple DES encryptions are being performed with multiple keys.

The next-to-last stage swaps the 32 bits on the left with the 32 bits on the right. Each of the remaining 16 iterations performs the same function but uses a different key. Specifically, the key at each iteration is generated from the key at the preceding iteration as follows. First a 56-bit permutation is applied to the key. Then the result is partitioned into two 28-bit blocks, each of which is independently rotated left by some number of bits. The combined result undergoes another permutation. Finally, a subset of 48 bits is used for the key at the given iteration.

The operation at each iteration is shown in Figure 11.24. The 64-bit input is divided into two equal portions denoted by L_{i-1} and R_{i-1} . The output generates two 32-bit blocks denoted by L_i and R_i . The left part of the output is simply equal to the right part of the input. The right part of the output is derived from the bitwise XOR of the left part of the input and a function of the right part of the input and the key at the given iteration.

The preceding algorithm simply breaks a long message into 64-bit blocks, each of which is independently encrypted using the same key. In this scheme DES is said to be operating in the **electronic codebook (ECB)** mode. This mode may not be secure when the structure of the message is known to the attacker.

Deficiency in the ECB mode can be removed by introducing dependency among the blocks. A simple way to introduce the dependency is to XOR the current plaintext block with the preceding ciphertext block. Such a scheme is shown in Figure 11.25 and is called **cipher block chaining (CBC)**. The first plaintext block is XORed with a given *initialization vector (IV)* that can be transmitted to the receiver in ciphertext for maximum security. CBC is generally the preferred approach for messages longer than 64 bits.

As alluded earlier, using a 56-bit key makes DES vulnerable to brute-force attack. One well-known improvement is called **triple DES**, which actually uses two keys, extending the overall key length to 112 bits. Choosing two keys instead of three keys

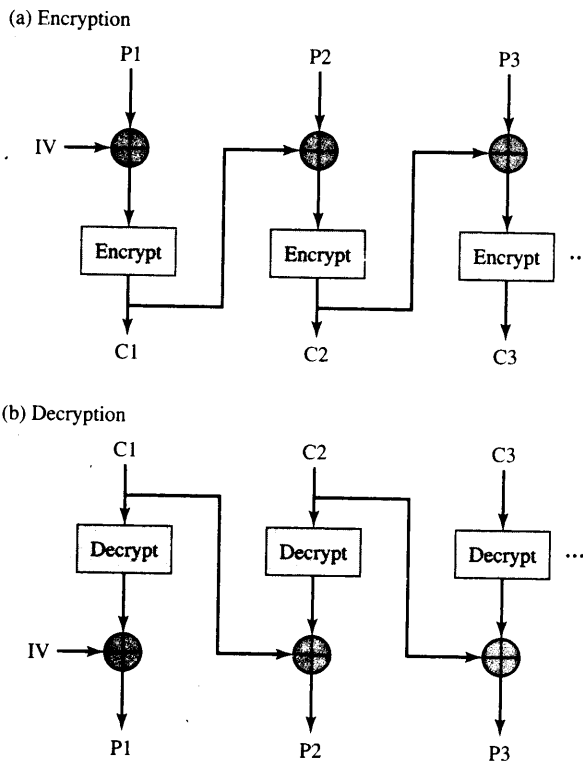


FIGURE 11.25 Cipher block chaining.

reduces the overhead significantly without compromising the security for commercial purposes. Triple DES performs the following encryption algorithm:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P))) \quad (11.1)$$

and the following decryption algorithm:

$$P = D_{K_1}(E_{K_2}(D_{K_1}(C))) \quad (11.2)$$

Note that the encryption algorithm uses an encryption-decryption-encryption (EDE) sequence rather than triple encryption EEE. This practice allows triple DES to talk to a single-key DES by making K_1 and K_2 the same, since $E_{K_1}(D_{K_1}(E_{K_1}(P))) = E_{K_1}(P)$.

In 1997 the National Institute of Standards and Technology (NIST) in the United States announced a public contest to select the successor of DES. In 2001 the Rijndael⁵ proposal (by Belgian cryptographers Rijmen and Daemen) was selected as the **Advanced Encryption Standard (AES)**. The Rijndael algorithm provides symmetric key encryption of 128-bit blocks of data with keys of 128, 192, or 256 bits in length. The algorithm can be implemented in software as well as very efficiently in hardware. The AES algorithm is much more secure than DES: the 56-bit DES key provides 7.2×10^{16} possible keys while the 128-bit AES key provides 3.4×10^{38} possible keys. Thus if

⁵Pronounced "Reign Dahl," "Rain Doll," or "Rhine Dahl."

a one DES key can be recovered in one second, then it will take 149 thousand billion years to crack an AES key!

11.3.2 RSA

The RSA algorithm (named after its inventors, Rivest, Shamir, and Adleman) is a widely accepted scheme for public key cryptography. It utilizes modular arithmetic and factorization of large numbers.

The public and private keys are generated based on the following rules:

1. Choose two large prime numbers p and q such that the product is equal to n . The plaintext P that is represented by a number must be less than n . In practice, n is a few hundred bits long.
2. Find a number e that is relatively prime to $(p - 1)(q - 1)$. Two numbers are said to be relatively prime if they have no common factors except 1. The public key consists of $\{e, n\}$.
3. Find a number d such that $de = 1 \pmod{(p - 1)(q - 1)}$. In other words, d and e are multiplicative inverses of each other modulo $((p - 1)(q - 1))$. The private key consists of $\{d, n\}$.

The RSA algorithm is based on the fact that if n , p , q , d , and e satisfy properties 1 to 3 above, then for any integer $P < n$ the following key property holds:

$$P^{de} \pmod{n} = P \pmod{n} \quad (11.3)$$

The RSA algorithm uses binary keys that are several hundred bits long, typically 512 bits. RSA takes a binary block of plaintext of length smaller than the key length and produces a ciphertext that is the same length of the key. Suppose that P is an integer that corresponds to a block of plaintext. RSA encrypts P as follows:

$$C = P^e \pmod{n} \quad (11.4)$$

The above calculation will yield an integer between 0 and n , and hence will require the same number of bits as the key.

To decrypt the ciphertext C , the RSA algorithm raises C to the power d and reduces the result modulo n :

$$C^d \pmod{n} = (P^e)^d \pmod{n} = P^{de} \pmod{n} = P \pmod{n} = P. \quad (11.5)$$

Thus we see that the aforementioned key property ensures that the plaintext can be recovered by the owner of the private key.

Why is the RSA algorithm secure? We know the public key, n and d , so can't we just determine the other factor e ? It turns out that factoring large integers n is very computationally intensive using currently available techniques.

EXAMPLE Using RSA

Let us consider the following simple example. Suppose we have chosen $p = 5$, and $q = 11$. Then $n = 55$, and $(p - 1)(q - 1) = 40$. Next find a number e that is relatively prime to 40, say, 7. The multiplicative inverse of 7 modulo 40 yields $d = 23$. From number theory we can find such a d only if e is relatively prime to $((p - 1)(q - 1))$. Now the public key is $\{7, 55\}$, and the private key is $\{23, 55\}$.

Suppose a message "RSA" is to be protected. For simplicity the message is represented numerically as 18, 19, 1 and uses three plaintexts: $P_1 = 18$, $P_2 = 19$, and $P_3 = 1$. The resulting ciphertexts are

$$\begin{aligned} C_1 &= 18^7 \bmod 55 = 17 \\ C_2 &= 19^7 \bmod 55 = 24 \\ C_3 &= 1^7 \bmod 55 = 1 \end{aligned} \quad (11.6)$$

The reader may verify that the decryption produces

$$\begin{aligned} 17^{23} \bmod 55 &= 18 \\ 24^{23} \bmod 55 &= 19 \\ 1^{23} \bmod 55 &= 1 \end{aligned} \quad (11.7)$$

How do we calculate modular arithmetic involving large numbers such as $17^{23} \bmod 55$? Fortunately, we can simplify the computation by using the following property:

$$(ab) \bmod n = ((a \bmod n)(b \bmod n)) \bmod n \quad (11.8)$$

As an example, we can write the first decryption as

$$17^{23} \bmod 55 = 17^{16+4+2+1} \bmod 55 = (17^{16} 17^4 17^2 17) \bmod 55 \quad (11.9)$$

Now $17^2 \bmod 55 = 14$. Then we continue with $17^4 \bmod 55 = (17^2)^2 \bmod 55 = 14^2 \bmod 55 = 4046 \bmod 55 = 31$. Similarly, $17^8 \bmod 55 = (17^4)^2 \bmod 55 = 31^2 \bmod 55 = 961 \bmod 55 = 26$, and $17^{16} \bmod 55 = (17^8)^2 \bmod 55 = 26^2 \bmod 55 = 676 \bmod 55 = 16$.

Finally, we can write

$$17^{23} \bmod 55 = (17^{16} 17^4 17^2 17) \bmod 55 = (16 \times 31 \times 14 \times 17) \bmod 55 = 18 \quad (11.10)$$

which is the original plaintext P_1 .

SUMMARY

Security protocols are required to deal with various threats that arise in communication across a network. These threats include replay attacks, denial-of-service attacks, and various approaches to impersonating legitimate clients or servers. We saw that security protocols build on cryptographic algorithms to provide privacy, integrity, authentication, and nonrepudiation services.

Secret key cryptography requires the sharing of the same secret key by two users. Secret key algorithms have the advantage that they involve modest computation. However the need for clients and servers to manage a different key for each association is too complicated. We saw that key distribution centers can help deal with the problem of managing keys in a secret key system.

Public key cryptography simplifies the problem of key management by requiring each user to have a private key that is kept secret and a public key that can be distributed to all other users. Public key cryptography nevertheless requires certification authorities that can vouch that a certain public key corresponds to a given user. Public key algorithms tend to be more computationally intensive than secret key algorithms. Consequently security protocols tend to use public key techniques to establish a master key at the beginning of a session, which is then used to derive a session key that can be used with a secret key method. The classic Diffie-Hellman exchange is used to establish a joint secret across a network. We saw how recent security protocol standards address weaknesses in the Diffie-Hellman exchange through the use of cookies and authentication.

Security protocols can be used at various layers of the protocol stack. The IP Security standards provide for extensions to IPv4 and IPv6 headers so that authentication, privacy, and integrity service can be provided across a public internet. Virtual private networks can be created using the tunneling capabilities of IPsec. We examined the Secure Sockets Layer and Transport Layer Security protocols which can provide secure connections over TCP, for application layer protocols such as HTTP. We also mentioned the important examples of PGP and Kerberos authentication service which demonstrate how security can be applied at the application layer or above.


Finally, we showed that security protocols can operate over various cryptographic algorithms as long as certain general requirements are met. We provided a brief discussion of two important cryptographic algorithms, DES and RSA. The subject of cryptographic algorithms is currently an area of intense research. The student is referred to several excellent texts on the subject that are listed in the references.

CHECKLIST OF IMPORTANT TERMS

| | |
|------------------------------------|--------------------------------------|
| Advanced Encryption Standard (AES) | Encapsulating Security Payload (ESP) |
| authentication header (AH) | encryption |
| certification authority (CA) | encryption header |
| challenge | firewall |
| cipher | hashed message authentication |
| cipher block chaining (CBC) | code (HMAC) |
| ciphertext | internet key exchange (IKE) |
| cryptography | IP Security (IPSEC) |
| Data Encryption Standard (DES) | key distribution center (KDC) |
| decryption | keyed MD5 |
| digital signature | message authentication code (MAC) |
| electronic codebook (ECB) | message digest 5 (MD5) algorithm |

| | |
|-----------------------------------|---------------------------------|
| nonce | secret key cryptography |
| plaintext | secure hash algorithm-1 (SHA-1) |
| private key | security association |
| public key | session |
| public key cryptography | substitution cipher |
| replay attack | transport mode |
| response | transposition cipher |
| Rivest, Shamir, and Adleman (RSA) | triple DES |
| algorithm | tunnel |
| secret key | tunnel mode |

FURTHER READING

-  Daemen, J. and V. Rijmen, *The Design of Rijndael*, Springer-Verlag, New York, 2002.
- Fluhrer, S., I. Mantin, and A. Shamir, "Weakness in the Key Scheduling Algorithm of RC4," *Proceedings of the Eighth Annual Workshop on Selected Areas of Cryptography*, pp. 1–24, August 2001, Toronto, Canada.
- Freier, A. O., P. Karlton, and P. C. Kocher, "The SSL Protocol, Version 3.0," Transport Layer Security Working Group, November 18, 1996.
- ISO/IEC 9594-8, "IT-OSI-The Directory: Authentication Framework," 1997.
- Kaufman, C., R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World* (2nd Ed.), Prentice Hall, Upper Saddle River, New Jersey, 2002.
- Khare, R. and S. Agranat, "Upgrading to TLS within HTTP/1.1," Network Working Group, June 22, 1999.
- Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.), John Wiley & Sons, 1996.
- Stallings, W., *Cryptography and Network Security: Principles and Practice* (3rd ed.), Prentice Hall, Upper Saddle River, New Jersey, 2002.
- Stubbenfield, A., J. Ioannidis, and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP," *Proceedings of Network and Distributed Systems Security Symposium*, ISOC, 2002.
- Yeager, N. J. and R. E. McGrath, *Web Server Technology: The Advanced Guide for World Wide Web Information Providers*, Morgan Kaufmann, San Francisco, 1996.
- Zimmerman, P. R., *The Official PGP User's Guide*, MIT Press, Cambridge, MA, 1995.
- Zwicky, E. D., S. Cooper, D. B. Chapman, and D. Russel, *Building Internet Firewalls* (2nd ed.), O'Reilly, Cambridge, 2000.
- RFC 2246, T. Dierks and C. Allen, "The TLS Protocol Version 1.0," January 1999.
- RFC 2401, S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," November 1998.
- RFC 2402, S. Kent and R. Atkinson, "IP Authentication Header," November 1998.
- RFC 2406, S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," November 1998.
- RFC 2408, D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," November 1998.
- RFC 2409, D. Harkins and D. Carrel, "The Internet Key Exchange," November 1998.
- RFC 2412, H. K. Orman, "The OAKLEY Key Determination Protocol," November 1998.

See our website for additional references available through the Internet.

PROBLEMS

- 11.1. Suppose that a certain information source produces symbols from a ternary alphabet $\{A, B, C\}$. Suppose also that the source produces As three times as frequently as Bs and Bs twice as frequently as Cs. Can you break the following ciphertext?
 (a) CBA
 (b) CBAACCBACBBCCCCBACC
- 11.2. Suppose that a certain information source produces symbols from a ternary alphabet $\{A, B, C\}$. Suppose that it is known that As always occur in pairs. Suppose that it is also known that an encryption scheme takes blocks of four symbols and permutes their order. Can you break the following ciphertext: BABACAABACACBAAB?
- 11.3. In authentication using a secret key, consider the case where the transmitter is the intruder.
 (a) What happens if the transmitter initiates the challenge?
 (b) How secure is the system if the receiver changes the challenge value once every minute.
 (c) How secure is the system if the receiver selects the challenge value from a set of 128 choices?
- 11.4. Consider a system where a user is authenticated based on an ID and password that are supplied by the transmitter in plaintext. Does it make any difference if the password and ID are encrypted? If yes, explain why? If no, how would you improve the system?
- 11.5. Compare the level of security provided by a server that stores a table of IDs and associated passwords as follows:
 (a) Name and password stored unencrypted.
 (b) Name and password stored in encrypted form.
 (c) Hash of name and password stored.
- 11.6. Explain why the processing required to provide privacy service is more complex than the processing required for authentication and for integrity.
- 11.7. Consider the following hashing algorithm. A binary block of length M is divided into subblocks of length 128 bits, and the last block is padded with zeros to a length of 128. The hash consists of the XOR of the resulting 128-bit vectors. Explain why this hashing algorithm is not appropriate for cryptographic purposes.
- 11.8. The *birthday problem* is a standard problem in probability textbooks, and it is related to the complexity of breaking a good hashing function.
 (a) Suppose there are n students in a class. What is the probability that at least two students have the same birthdate? *Hint:* Find the probability that all students have a different birthdate.
 (b) In part (a) for what value of n is the probability that at least two students have the same birthdate approximately $\frac{1}{2}$?
 (c) Suppose we use a 64-bit hash function. If we obtain the hash outputs for n distinct messages, what is the probability that no two hashes are the same? For what value of n the probability of finding at least one matching pair $\frac{1}{2}$?
 (d) Concoct a digital signature scenario in which the ability to find a pair of messages that lead to the same hash output would be useful to an unscrupulous person.

- 11.9.** Consider the operation of the key distribution center: the KDC sends $E_{K_A}(K_{AB})$ and $E_{K_B}(K_{AB})$ to host A; Host A in turn sends the “ticket” $E_{K_B}(K_{AB})$ to host B. Explain why the term *ticket* is appropriate in this situation.
- 11.10.** Suppose a directory that stores certificates is broken into, and the certificates in the directory are replaced by bogus certificates? Explain why users will still be able to identify these certificates as bogus.
- 11.11.** Suppose that KDC A serves one community of users and KDC B serves a different community of users. Suppose KDC A and KDC B establish a shared key K_{AB} . Develop a method to enable a user α from KDC A to obtain a shared key with a user β from KDC B. Note that KDCs A and B must participate in the process of establishing the shared key $K_{\alpha\beta}$.
- 11.12.** Consider a highly simplified Diffie-Hellman exchange in which $p = 29$ and $g = 5$. Suppose that user A chooses the random number $x = 3$ and user B chooses the number $y = 7$. Find the shared secret K .
- 11.13.** Consider a firewall consisting of a packet-filtering router.
- Explain how this firewall could be set up to allow in only HTTP requests to a specific server.
 - Suppose an outside intruder attempts to send packets with forged internal addresses, which presumably are allowed access to certain systems. Explain how a packet-filtering router can detect these forged packets.
 - Explain how packets from a given remote host can be kept out.
 - Explain how inbound mail to specific mail servers is allowed in.
 - Explain how inbound Telnet service can be blocked.
- 11.14.** The ICMP “Host Unreachable” error message is sent by a router to the original sender when the router receives a datagram that it cannot deliver or forward. Suppose that the router in question is a packet-filtering router. Discuss the pros and cons of having the router return an ICMP message from the following viewpoints: processor load on the router, traffic load in the network, susceptibility to denial-of-service attacks, and disclosure of filtering operation to intruders.
- 11.15.** Compare the following two approaches to operation of a packet-filtering router: discard everything that is not expressly permitted versus forward everything that is not expressly prohibited.
- Which policy is more conservative?
 - Which policy is more visible to the users?
 - Which policy is easier to implement?
- 11.16.** Identify the fields in the IPv4 header that cannot be covered by the authentication header. What can be done to protect these fields?
- 11.17.** Explain the benefits the authentication header in IPSec brings to the operation of a packet-filtering router.
- 11.18.** Explain why a packet-filtering router should reassemble a packet that has been fragmented in the network and check its authentication header, instead of forwarding the fragments to the destination.

- 11.19. (a) Explain how the use of cookies thwarts a denial-of-service attack in the Diffie-Hellman exchange.
 (b) Explain how authentication thwarts the man-in-the-middle attack.
- 11.20. Consider Figure 11.9a where two hosts communicate directly over an internet. Explain the rationale for the following uses of AH and ESP modes in IPSec. Show the resulting IP packet formats.
 (a) Transport mode: AH alone.
 (b) Transport mode: ESP alone.
 (c) Transport mode: AH applied after ESP.
 (d) Tunnel mode: AH alone.
 (e) Tunnel mode: ESP alone.
- 11.21. Explain how IPSec can be applied in the exchange of routing information, for example, OSPF.
- 11.22. Consider the arrangement in Figure 11.9b where two internal networks are protected by firewalls. Explain the rationale for establishing a tunnel between gateways using AH and having the end hosts use ESP in transport mode.
- 11.23. Consider the arrangement in Figure 11.9c where a mobile host communicates with an internal host across a firewall. Explain the rationale for establishing an AH tunnel between the remote host and the firewall and ESP in transport mode between the two hosts.
- 11.24. When is IPSec appropriate? When is SSL/TLS appropriate?
- 11.25. Suggest some SSL/TLS situations in which it is appropriate to authenticate the client.
- 11.26. HTTP over SSL uses URLs that begin with https: and use TCP port 443, whereas HTTP alone uses URLs that begin with http: and use TCP port 80. Explain how a secure and unsecured mode using the same port number 80 may be possible by introducing an upgrade mechanism in HTTP.
- 11.27. Suppose DES is used to encrypt a sequence of plaintext blocks $P_1, P_2, P_3, \dots, P_i, \dots, P_N$ into the corresponding ciphertext blocks $C_1, C_2, C_3, \dots, C_i, \dots, C_N$.
 (a) If block C_i is corrupted during transmission, which block(s) will not be decrypted successfully using the ECB mode?
 (b) Repeat (a) using the CBC mode.
- 11.28. Using the RSA algorithm, encrypt the following:
 (a) $p = 3, q = 11, e = 7, P = 12$
 (b) $p = 7, q = 11, e = 17, P = 25$
 (c) Find the corresponding d s for (a) and (b) and decrypt the ciphertexts.

Multimedia Information

In Chapters 1 and 3 we discussed the trend toward communication networks that can handle a wide range of information types including multimedia. This chapter provides a more detailed presentation of the formats, the bit rates, and other properties of important types of information including text, speech, audio, facsimile, image, and video. We show that the representation of audio and video information can require higher bit rates and lower delays than have been traditional in packet networks. The protocols introduced in Chapter 10 will provide packet networks with the responsiveness to meet these requirements and the ability to set up a wide variety of multimedia sessions.

This chapter is organized as follows:

1. *Lossless data compression.* We present techniques that reduce the number of bits required to represent a file of information under the condition that the original file can be recovered in its exact form. We discuss how existing standards apply lossless data compression to text and other computer files, as well as to facsimile.
2. *Techniques for compression of analog signals.* We present techniques that reduce the number of bits required to represent a block or stream of digitized analog signals under the condition that the signal can be recovered approximately but within some level of fidelity. We discuss the application of these techniques to speech and audio signals. In particular we introduce the MPEG MP3 compression standards for audio.
3. *Techniques for compression of image and video signals.* We present techniques for the compression of individual as well as sequences of image information. We introduce the MPEG video coding standard.

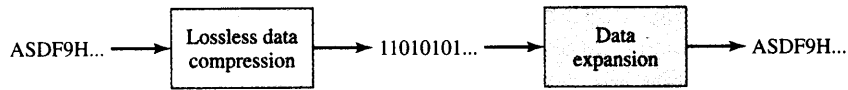


FIGURE 12.1 Lossless data compression.

12.1 LOSSLESS DATA COMPRESSION

In lossless data compression we are given a block of digital information, and we are interested in obtaining an efficient digital, usually binary, representation. Examples of such a block of information are any computer files generated by applications, such as word processors, spreadsheets, and drawing programs. Lossless data compression techniques are also used by modems and facsimile machines to reduce the time to transmit files and by operating systems and utility programs to reduce the amount of disk space required to store files.

We will view a block of information as a sequence of symbols from some alphabet as shown in Figure 12.1. The objective of lossless data compression is to map the original information sequence into a string of binary digits so that (1) the average number of bits/digital symbol is small and (2) the original digital information sequence can be recovered exactly from encoded binary stream. We will assess the performance of any given lossless data compression code by the average number of encoded bits/symbol.

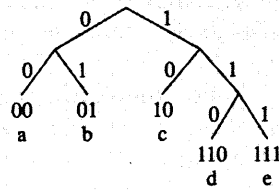
12.1.1 Huffman Codes

The Morse code uses the basic principle for achieving efficient representations of information: frequently occurring symbols should be assigned short codewords, and infrequent symbols should be assigned longer codewords. In this manner the average number of bits/symbol is reduced. In 1954 Huffman invented an algorithm for identifying the code that is optimal in the sense of minimizing the average number of bits/symbol. A **Huffman code** segments the original sequence of symbols into a sequence of fixed-length blocks. Each block is assigned a variable-length binary stream called a **codeword**. The codewords are selected so that no codeword is a prefix of another codeword. The set of all codewords then form the terminal nodes of a binary tree such as the one shown in Figure 12.2.

Assume

- Five-symbol information source: {a, b, c, d, e}.
- Symbol probabilities: {1/4, 1/4, 1/4, 1/8, 1/8}.

| Symbol | Codeword |
|--------|----------|
| a | 00 |
| b | 01 |
| c | 10 |
| d | 110 |
| e | 111 |



aedbbad ... mapped into 00 111 110 01 01 00 110 ... 17 bits
 Note: Decoding done without commas or spaces.

FIGURE 12.2 Huffman coding example.

Before discussing the Huffman procedure for obtaining the best code, we will discuss a simple example to demonstrate the encoding and decoding procedures. For simplicity we will consider encoding individual symbols, that is, blocks of length 1. This example assumes that our information source generates a stream of symbols from the alphabet {a, b, c, d, e} and that the symbols occur with respective probabilities {1/4, 1/4, 1/4, 1/8, 1/8}. Figure 12.2 shows a binary code that assigns two-bit and three-bit codewords to the symbols. The figure also shows the binary string that results from encoding the sequence of symbols aedbbad. . . . The binary tree code in this figure shows how the codewords correspond to the five terminal nodes of the tree.

The tree in Figure 12.2 can be used to demonstrate the general decoding procedure. At the beginning of the decoding, we begin at the top node of the tree. Each encoded bit determines which branch of the tree is followed. The sequence of encoded bits then traces a path down the tree until a terminal node is reached. None of the intermediate nodes prior to this terminal node can correspond to a codeword by the design of the code. Therefore, as soon as a terminal node is reached, the corresponding symbol can be output. The decoder then returns to the top node of the tree and repeats this procedure. This technique, while efficiently compacting the number of binary bits, increases the effect of transmission errors. For example, if the 00 for “a” incurs an error giving 11, the result will be not only a change in the letter but also an error in the alignment of sets of data bits with letters.

Let $\ell(s)$ be the length of the codeword assigned to symbol s . The performance of the code is given by the average number of encoded bits/symbol, which is given by

$$\begin{aligned} E[\ell] &= \ell(a)P[a] + \ell(b)P[b] + \ell(c)P[c] + \ell(d)P[d] + \ell(e)P[e] \\ &= 2(.25) + 2(.25) + 2(.25) + 3(.125) + 3(.125) \\ &= 2.25 \text{ bits/symbol} \end{aligned} \tag{12.1}$$

The simplest code to use for this information source would assign codewords of equal length to each symbol. Since the number of codewords is five, three-bit codewords would be required. The performance of such a code is three bits/symbol. Thus, for example, a file consisting of 10,000 symbols would produce 30,000 bits using three-bit codewords and an average of 22,500 bits using the Huffman code.

The extent to which compression can be achieved depends on the probabilities of the various symbols. For example, suppose that a source produces symbols with equal probability. Clearly the best way to code the symbols is to give them codewords of equal length to the extent possible. In particular, if the source produces symbols from an alphabet of size 2^m , then the best code simply assigns an m -bit codeword to each symbol. No other code can produce further compression. The Shannon entropy, which is introduced in the next section, specifies the best possible compression performance for a given information source.

The design of the Huffman code requires knowledge of the probabilities of the various symbols. In certain applications these probabilities are known ahead of time, so the same Huffman code can be used repeatedly. If the probabilities are not known, codes that can adapt to the symbol statistics “on the fly” are preferred. Later in this section we present the Lempel-Ziv adaptive code.

◆ HUFFMAN CODE ALGORITHM AND THE SHANNON ENTROPY

We are now ready to discuss the Huffman algorithm. We assume that the probabilities of the symbols are known. The algorithm starts by identifying the two symbols with the smallest probabilities. It can be shown that the best code will connect the terminal nodes of these two symbols to the same intermediate node. These two symbols are now combined into a new symbol whose probability is equal to the sum of the two probabilities. You can imagine the two original symbols as being placed inside an envelope that now represents the combined symbol. In effect we have produced a new alphabet in which the two symbols have been replaced by the combined symbol. The size of the new alphabet has now been reduced by one symbol. We can again apply the procedure of identifying the two symbols with the smallest probabilities from the reduced alphabet and combining them into a new symbol. Each time two symbols are combined, we connect the associated nodes to form part of a tree. The procedure is repeated until only two symbols remain. These are combined to form the root node of the tree.

The Huffman procedure is demonstrated in Figure 12.3a where we consider a source with five symbols {a, b, c, d, e} with respective probabilities {.50, .20, .15, .10, .05}. The first step of the Huffman algorithm combines symbols d and e to form a new symbol, which we will denote by (de) and which has combined probability .15. The terminal node for symbols d and e are combined into the intermediate node denoted by 1 in Figure 12.3a. The new alphabet now consists of the symbols {a, b, c, (de)} with probabilities {.50, .20, .15, .15}. The second step of the Huffman algorithm combines the symbols c and (de) into the new symbol (c(de)) with combined probability of .3. These are combined into intermediate node 2 in Figure 12.3a. The third step of the algorithm combines symbols b and (c(de)) into (b(c(de))). The final step combines the two remaining symbols a and (b(c(de))) to form the root node of the tree.

In Figure 12.3b we have rearranged the tree code obtained from the Huffman algorithm and obtained the codewords leading to the terminal nodes. This is done by starting at the root node and assigning a 0 to each left branch and a 1 to each right branch. The resulting code is shown in Figure 12.3b. Let $\ell(s)$ be the length of the codeword assigned to symbol s . The performance of the code is given by the average

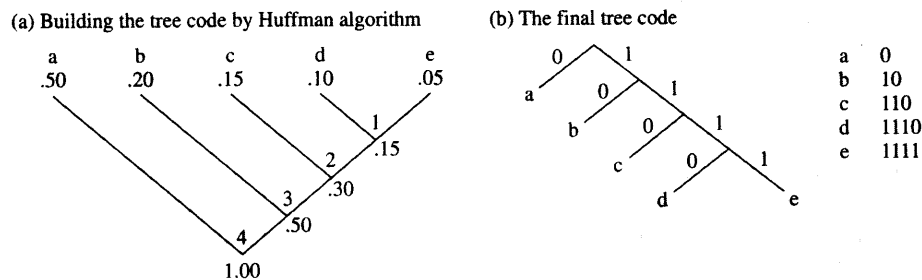


FIGURE 12.3 Building a Huffman tree code.

number of encoded bits/symbol, which is given by

$$\begin{aligned} E[\ell] &= \ell(a)P[a] + \ell(b)P[b] + \ell(c)P[c] + \ell(d)P[d] + \ell(e)P[e] \\ &= 1(.50) + 2(.20) + 3(.15) + 4(.10) + 4(.05) \\ &= 1.95 \text{ bits/symbol} \end{aligned} \quad (12.2)$$

The simplest code that could have been used for this information source would assign three-bit codewords to each symbol. Thus we see that the Huffman code has resulted in a representation more efficient than the simple code.

We indicated above that in general Huffman coding deals with blocks of n symbols. To obtain the Huffman codes where n is greater than 1, we arrange all possible n -tuples of symbols and their corresponding probabilities. We then apply the Huffman algorithm to this superalphabet to obtain a binary tree code. For example, the $n = 2$ code in Figure 12.3 would consider the alphabet consisting of the 25 pairs of symbols {aa, ab, ac, ad, ae, ba, bb, . . . , ea, eb, ec, ed, ee}. In general, performance of the code in terms of bits/symbol will improve as the block length is increased. This of course, is accompanied by an increase in complexity.

In 1948 Shannon addressed the question of determining the best performance attainable in terms of encoded bits/symbol for any code of any block length. He found that the best performance was given by the entropy function, which depends on the probabilities of sequences of symbols. Suppose that the symbols are from the alphabet $\{1, 2, 3, \dots, K\}$ with respective probabilities $\{P[1], P[2], P[3], \dots, P[K]\}$. If the sequence of symbols are statistically independent, the **entropy** is given by

$$\begin{aligned} H &= - \sum_{k=1}^K P[k] \log_2 P[k] \\ &= -P[1] \log_2 P[1] - P[2] \log_2 P[2] \dots - P[K] \log_2 P[K] \end{aligned} \quad (12.3)$$

Shannon proved that no code can attain an average number of bits/symbol smaller than the entropy. In Equation 12.3 the logarithms are taken to the base 2, which can be obtained as follows:

$$\log_2 x = \ln x / \ln 2 \quad (12.4)$$

where $\ln x$ is the natural logarithm.

For the example in Figure 12.3, the entropy is given by

$$\begin{aligned} H &= (-.50 \ln .50 - .20 \ln .20 - .15 \ln .15 - .10 \ln .10 - .05 \ln .05) / \ln 2 \\ &= 1.923 \end{aligned} \quad (12.5)$$

This result indicates that the code obtained in Figure 12.3 is sufficiently close to the best attainable performance and that coding of larger block lengths is unnecessary. If the performance of the code had differed significantly from the entropy, then larger block lengths could be considered.

As an additional example consider the code in Figure 12.2. It can be easily shown that the Huffman algorithm will produce this code. The average number of bits/symbol for this code is 2.25 bits. Furthermore, the entropy of this code is also 2.25 bits/symbol.

In other words, this code attains the best possible performance, and no increase in block length will yield any improvements.

The entropy formula can also be used as a guideline for obtaining good codes. For example, suppose that a source has a K symbol alphabet and that symbols occur with equal probability, that is, $1/K$. The entropy is then given by

$$H = - \sum_{k=1}^K P[k] \log_2 P[k] = - \sum_{k=1}^K \frac{1}{K} \log_2 \frac{1}{K} = \log_2 K \quad (12.6)$$

In the special case where $K = 2^m$, we have $H = \log_2 2^m = m$ bits/symbol. Note that we can assign each symbol an m -bit codeword and achieve the entropy. Thus in this case this simple code achieves the best possible performance. This result makes intuitive sense, since the fact that the symbols are equiprobable suggests that they should have the same length. More generally, by comparing the expression for the entropy and the expression for the average number of bits/symbol, we see that the length of the k th symbol can be identified with the term $-\log_2 / P[k]$. This suggests that a good code will assign to a symbol that has probability $P[k]$ a binary codeword of length $-\log_2 / P[k]$. For example, if a given symbol has probability $1/2$, then it should be assigned a one-bit codeword.

12.1.2 Run-Length Codes

In many applications one symbol occurs much more frequently than all other symbols, as shown in Figure 12.4. The sequence of symbols produced by such information sources consist of many consecutive occurrences of the frequent symbol, henceforth referred to as **runs**, separated by occurrences of the other symbols. For example, the files corresponding to certain types of documents will contain long strings of blank characters. Facsimile information provides another example of where the scanning process produces very long strings of white dots separated by short strings of black dots. Run-length coding is a very effective means of achieving lossless data compression for these types of information sources. Instead of breaking the sequence of symbols into fixed-length blocks and assigning variable-length codewords to these blocks, a **run-length code** parses the sequence into variable-length strings consisting of consecutive occurrences of the common symbol and the following other symbol. The codeword for each run consists of a binary string to specify the length of the run, followed by a string that specifies the terminating symbol. Now if very short binary codewords are used to specify the length of very long runs, then it is clear that huge compression factors are being achieved. In the remainder of this section we focus on the run-length coding of binary sources that are typified by facsimile.

- "Blank" in strings of alphanumeric information
 ----- \$5 ---- 3 ----- \$2 ---- \$3 -----

- "0" (white) and "1" (black) in fax documents
 ○○○○○●○○○○○○○○○○●○

FIGURE 12.4 Run-length coding—Introduction.

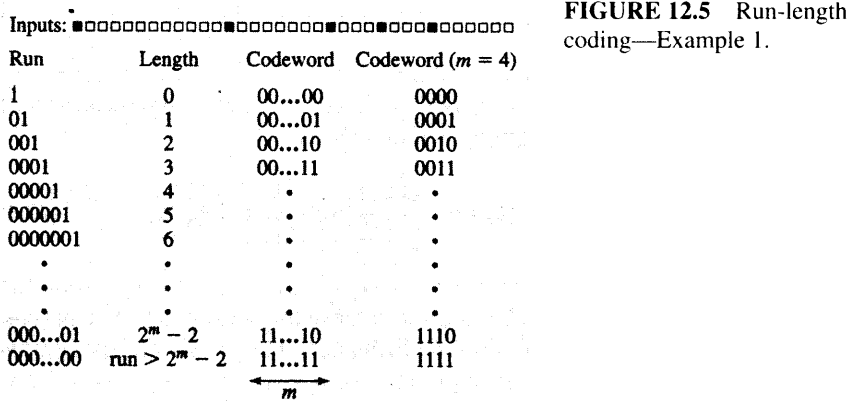


FIGURE 12.5 Run-length coding—Example 1.

Figure 12.5 describes the simplest form of run-length coding for binary sources where 0s (white dots) are much more frequent than 1s (black dots). The sequence of binary symbols produced by the information source is parsed into runs shown in the left column. The encoder carries out this parsing simply by counting the number of 0s between 1s and producing an m -bit binary codeword that specifies the number of 0s. When two consecutive 1s occur, we say that a run of length 0 has occurred. The figure shows that the maximum complete run that can be encoded has length $2^m - 2$. Thus when $2^m - 1$ consecutive 0s are observed, the encoder must terminate its count. The code in Figure 12.5 has the encoder output a codeword consisting of all 1s that tells the decoder to output $2^m - 1$ consecutive 0s; the encoder resets its counter to 0 and starts a new run.

Figure 12.6 shows a string of 137 consecutive binary symbols that are encoded into four-bit codewords, and then decoded to obtain original sequence. In the example the 137 original binary symbols are compacted into 44 bits. As before, the performance of the lossless data compression coding scheme is given by the average number of encoded bits per source symbol, which is given by $m/E[R]$ where m is the number of bits in the codeword and $E[R]$ is the average number of symbols encoded in each

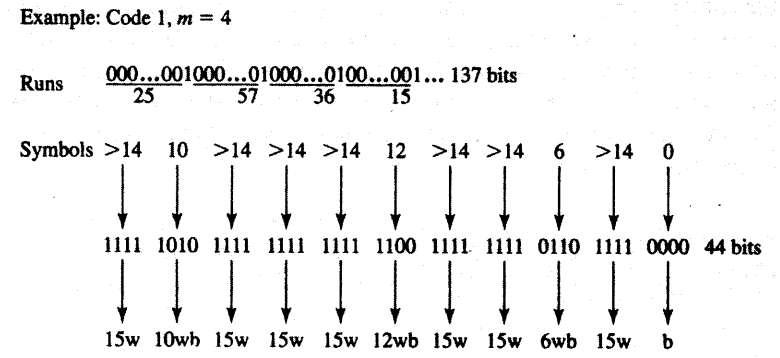


FIGURE 12.6 Run-length encoding and decoding.

Example: Code 2, $m = 4$

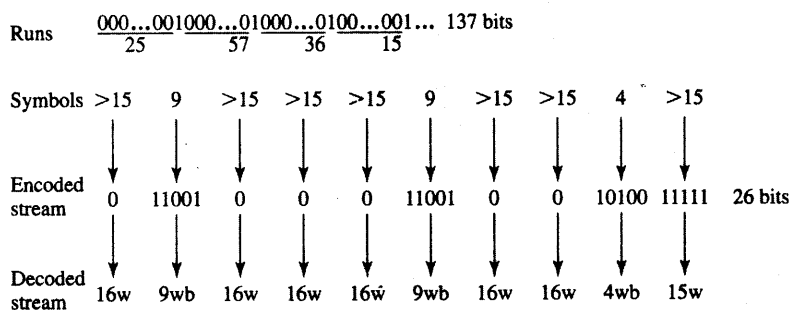


FIGURE 12.8 Variable-to-variable run-length coding using Huffman coding.

average number of symbols encoded in each run. Suppose again that the probability of a zero is 0.96. A numerical calculation then gives $E[\ell] = 2.92$ and $E[R] = 12$, for a compression ratio of $12/2.92 = 4.11$.

FACSIMILE CODING STANDARDS

Run-length coding forms the basis for the coding standards that have been developed for facsimile transmission. In facsimile a black and white image is scanned and converted into a rectangular array of dots called **pixels**. Each pixel corresponds to a measurement made at a given point in the document, and each pixel is assigned a value 0 (for white) or 1 (for black) according to the measured intensity. The International Telecommunications Union (ITU) standard defines several options for encoding facsimile images. The standards assume a scanning resolution of 200 dots/inch in the horizontal direction and 100 dots/inch in the vertical direction. For an 8.5- \times -11-inch standard North American document, the scanning process will produce $8.5 \times 200 \times 11 \times 100 = 1,870,000$ pixels. Because each pixel is represented by one bit, each page corresponds to 235 kilobytes where a byte consists of eight bits. Using a modem that transmits at a speed of 10,000 bits/second, we see that the transmission of each page will require about 3 minutes. Clearly lossless data compression is desirable in this application.

The ITU considered a set of eight documents, listed in Table 12.1, in defining the facsimile coding standards. These documents are a business letter, a simple hand-drawn circuit diagram, an invoice form, a page with dense text, a page from a technical paper, a graph, a page of dense Japanese text, and a page with simple graphics and handwriting. After scanning and prior to compression, each document required 256 kilobytes. (The ITU was considering A4 standard-size paper that is slightly larger than the 8.5- \times -11-inch North American standard.) Four groups of standards are defined for facsimile by the ITU. The Group I and Group II standards are analog in nature. The Group II column in the table corresponds to a scanned but uncompressed version of the Group II standard. Groups III and IV use lossless data compression techniques.

The Group III standard defined by ITU uses a so-called one-dimensional modified Huffman coding technique that combines run-length coding and Huffman coding, as

TABLE 12.1 Compression results for ITU coding standards.

| CCITT | G-II | G-III | G-IV | Time for G-IV (9.6 kbps) |
|------------------------------------|------|-------|-------|-----------------------------|
| Business letter | 256K | 17K | 10K | 8.33 sec |
| Circuit diagram | 256K | 15K | 5.4K | 4.50 sec |
| Invoice | 256K | 31K | 14K | 11.67 sec |
| Dense text | 256K | 54K | 35K | 29.17 sec |
| Technical paper | 256K | 32K | 16K | 13.33 sec |
| Graph | 256K | 23K | 8.3K | 6.92 sec |
| Dense Japanese text | 256K | 53.5K | 34.6K | 28.83 sec |
| Handwriting and simple graphics | 256K | 26K | 10K | 8.33 sec |
| Average | 256K | 31.4K | 16.6K | |
| Compression ratio | 1 | 8.2 | 15.4 | |
| Maximum compression ratio | 17.1 | 47.4 | | |

Note: Documents scanned at 200×100 pixels/square inch. G-IV intended for documents scanned at 400×100 pixels/square inch and ISDN transmission at 64 kbps.

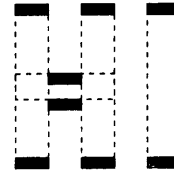
shown in Figure 12.9a. The ITU standards committee measured the statistics of runs of white symbols followed by runs of black symbols for the combined set of test documents. In addition to the symbols specifying the run lengths, special characters such as end-of-line and Escape need to be considered. Using this information, it is then possible to apply the Huffman algorithm to obtain a code for encoding the run length. However, because of the long runs that can occur, the resulting Huffman code can be quite complex. As a result, the ITU committee developed a modified, but simpler to implement, Huffman code to encode the run lengths. Because the statistics of the white runs and black runs differ, the Group III standard uses different Huffman codes to encode the white runs and black runs.

The Group III column in Table 12.1 shows the number of kilobytes for each test document after compression using Group III encoding scheme. The business letter and the circuit diagram, which contained a large number of white pixels, are compressed by factors of 15 or so. However, the documents with denser text are only compressed by factors of 5 or so. Overall the compression ratio for the entire set of documents is about 8. To the extent that these documents are typical, compression ratios of this order can be expected. Table 12.1 shows the transmission times for the various documents using the CCITT V.29 fax modem speed of 9.6 kbps. The later standard V.17 allows for a speed of 14.4 kbps.

(a) Huffman code is applied to white runs and black runs



(b) Encode differences between consecutive lines

**FIGURE 12.9** Facsimile coding standards.

The Group IV standard exploits the correlation between adjacent scanned lines. In black-and-white documents, black areas form connected regions so that if a given pixel is black in a given scanned line, the corresponding pixel in the next line is very likely to also be black. By taking the modulo 2 sum of pixels from adjacent lines, we will obtain 1s only at the pixel locations where the lines differ as shown in Figure 12.9b. Only a few 1s can be expected, since adjacent lines are highly correlated. In effect, this processing has increased the number and the length of the white runs leading to higher compression performance. The Group IV standard developed by the ITU uses a modified version of this approach in which Huffman codes encode the position and number of pixel changes relative to the previous scanned line. In Table 12.1 we see that the Group IV standard achieves a compression ratio approximately double that of Group III. Because a scanned line is encoded relative to a previous line, an error in transmission can cause an impairment in all subsequent lines. For this reason the ITU standard recommends that the one-dimensional technique be applied periodically and that the number of lines encoded using the two-dimensional technique be limited to some predefined number.

The scanning density of the original ITU standards has long been superseded by laser and inkjet printing technology. The ITU has modified the standard to allow for resolutions of 200, 300, and 400 pixels per inch. As the scanning resolution is increased, we can expect that the lengths of the horizontal runs will increase and that the correlation between adjacent vertical lines will increase. Hence it is reasonable to expect to attain significantly higher compression ratios than those shown in Table 12.1.

12.1.3 Adaptive Data Compression Codes

The techniques so far require that we know the probability of occurrence of the information symbols. In certain applications these probabilities can be estimated once, and it is reasonable to assume that they will remain fixed. Examples include facsimile and certain types of documents. On the other hand, the statistics associated with certain information sources either are not known ahead of time or vary with time. For this reason it is desirable to develop **adaptive lossless data compression** codes that can achieve compaction in these types of situations.

The most successful adaptive lossless data compression techniques were developed by Ziv and Lempel in the late 1970s. In run-length coding, the information stream is parsed into black runs and white runs, and the runs are encoded using a variable-length code. The Lempel-Ziv approach generalizes the principle behind run-length coding and parses the information stream into *strings of symbols* that occur frequently and encodes them using a variable-length code. In the Lempel-Ziv algorithm the encoder identifies repeated patterns of symbols and encodes in the following manner. Whenever a pattern is repeated in the information sequence, the pattern is replaced by a pointer to the first occurrence of the pattern and a value that indicates the length of the pattern. The algorithm is adaptive in that the frequently occurring patterns are automatically identified by the encoding process as more and more of the symbol sequence is processed.

In Figure 12.10, we give an example of the algorithm for a rhyme from Dr. Seuss [1963]. Most beginning readers have high redundancy, since that is one of the main

“All tall We all are tall. All small We all are small.”

· Can be mapped into

“All_ta[2, 3]We_[6, 4]are[4, 5]_[1, 4]sm[6, 15][31, 5].”

FIGURE 12.10 Lossless data compression using the Lempel-Ziv algorithm.

characteristics that make the reader easy to read. The first six characters, All_ta, are left unchanged, since they do not contain a repeated pattern. The next three symbols, ll_, are seen to have occurred before, so the marker [2,3] is used to indicate the decoder should refer back to the second character and reproduce the next three characters. The next three characters, We_, are left unchanged, but the following four characters, all_, are seen to have occurred starting with character 6. The next three characters, are, are left unchanged, but the next five characters, _tall, are seen to have occurred starting with character 4. The next two characters, .., are left unchanged, and the following four characters, All_, are seen to have occurred starting with character 1. The next two characters, sm, are unchanged, and we then hit the jackpot by finding that the next 15 characters, all_We_all_are_, occurred starting with character 6. Finally, the last word is seen to have occurred starting with character 31. The final period completes the encoding. In this example the original sequence consisted of 53 ASCII characters, and the compressed sequence consists of 29 ASCII characters, where we assume that each number in the pointer takes one ASCII character. Isn't this fun? In the problem section you get a chance to decode another one of our favorite Dr. Seuss rhymes.

ERROR CONTROL AND LOSSLESS DATA COMPRESSION

Lossless data compression is achieved by removing a redundancy in the sequence of symbols used to represent a given set of information. The reduced redundancy implies that the resulting compressed set is more vulnerable to error. As shown for the case of two-dimensional facsimile coding, individual errors can propagate and cause significant impairments in the recovered image. Similarly, errors introduced in the encoded sequence that results from Huffman coding can result in complete loss of the information. The effect of errors on information that has been compressed can be reduced through the use of error-correction techniques and through the insertion of synchronization information that can limit error propagation.

The above version of the Lempel-Ziv algorithm requires the encoder to search backward through the entire file to identify repeated patterns. The algorithm can be modified by restricting the search to a window that extends from the most recent symbol to some predetermined number of prior symbols. The pointer no longer refers to the first occurrence of the pattern but rather to the most recent occurrence of the pattern. Another possible modification of the Lempel-Ziv algorithm involves limiting the search to patterns contained in some dictionary. The encoder builds this dictionary as frequently occurring patterns are identified.

Adaptive lossless data compression algorithms are used widely in information transmission and storage applications. In transmission applications, lossless data

compression can reduce the number of bits that need to be transmitted and hence the total transmission time. In storage applications lossless data compression techniques may reduce the number of bits required to store files and hence increase the apparent capacity of a given storage device. Three types of lossless data compression techniques have been implemented in conjunction with modems. The MMP5 protocol uses a combination of run-length coding and adaptive Huffman coding. The MMP7 protocol exploits the statistical dependence between characters by applying context-dependent Huffman codes. The V.42bis ITU modem standard use the Lempel-Ziv algorithm. The Lempel-Ziv algorithm has also been implemented in storage applications such as the UNIX COMPRESS command. The Lempel-Ziv algorithm yields compression ratios ranging from 2–6 in file-compression applications.

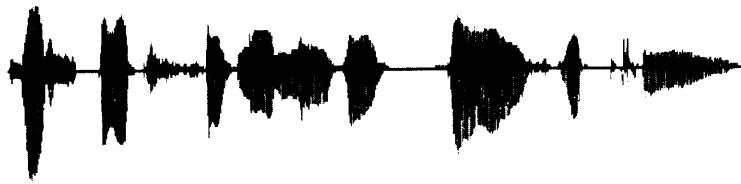
12.2 COMPRESSION OF ANALOG SIGNALS

We now consider the compression of analog information. In Chapter 3 we indicated that analog information is characterized by the fact that its exact representation requires an infinite number of bits. Examples of analog information include speech and audio signals that consist of the continuous variation of amplitude versus time. Another example of analog information is image information that consists of the variation of intensity over a plane. Video and motion pictures are yet another example of analog information where the variation of intensity is now over space and time. All of these signals can assume a continuum of values over time and/or space and consequently require infinite precision in their representation. All of these signals are also important in human communications and are increasingly being incorporated into a variety of multimedia applications.

Lossy data compression involves the problem of representing information within some level of approximation. In general, data compression is required to represent analog signals using only a finite number of bits. Therefore, a figure of merit for a lossy data compression technique is the quality, fidelity, or degree of precision that is achieved for a given number of representation bits. In Chapter 3 we presented the sampling and quantization techniques for digitizing analog information. In this section we present a number of techniques for achieving additional compression of this analog information. These techniques include adaptive quantization, predictive coding, and transform coding. We also discuss how these techniques are used in standards for coding speech, audio, image, and video signals.

12.2.1 Adaptive Quantizers

The derivation for the performance of the uniform quantizers in Chapter 3 assumed that we know the exact dynamic range $-V$ to V in which the signal values will fall. Frequently the system does not have sufficient control over signal levels to ensure that this is the case. For example, the distance a user holds a microphone from his or her mouth will affect the signal level. Thus if a mismatch occurs between the signal level for



The speech signal level varies with time

FIGURE 12.11 Variation of signal level for speech.

which the system has been designed and the actual signal level, then the performance of the quantizer will be affected. Suppose, for example, that the quantizer is designed for the dynamic range $-V$ to V and the actual signal level is in the range $-V/2$ to $V/2$. Then in effect we are using only half the signal levels, and it is easy to show that the SNR will be 6 dB less than if the signal occupied the full dynamic range. Conversely, if the actual signal level exceeds the dynamic range of the quantizer, then the larger-magnitude samples will all be mapped into the extreme approximation values of the quantizer. This type of distortion is called **clipping** and in the cassette recording systems that you are surely familiar with is indicated by a red light in the signal-level meter.

Figure 12.11 shows the speech waveform for the following sentence: The speech signal level varies with time. The waveform is about 3 seconds long and was sampled at a rate of 44 kHz, so it consists of approximately 130,000 samples. Large variations in signal levels can be observed.

One way of dealing with variations in signal level is to actually measure the signal levels for a given time interval and to multiply the signal values by a constant that maps the values into the range for which the quantizer has been designed. The constant is then transmitted along with the quantizer values. These types of quantizers are called **adaptive quantizers**. A passive way of dealing with variations in the signal level is to use **nonuniform quantizers** where the quantizer intervals are roughly proportional to the signal level. The *companders* used for telephone speech are an example of this.

12.2.2 Predictive Coding

In Chapter 3 we introduced **pulse code modulation (PCM)**, the basic technique for the conversion of analog signals such as voice and video to digital form. The quantizers that were introduced in Section 3.3 simply convert the samples of an analog signal into digital form. They do not deal with statistical dependencies between samples. For example, in sections where the signal is changing slowly, consecutive samples will tend to have values that are close to each other, as shown in Figure 12.12. In another example, the signal is approximately periodic, and longer-term dependencies between sample values can be observed. Figure 12.13 shows the waveform for the sound corresponding to the English vowel sound “ae,” as in the word *cat*. The long-term dependency between sample values is evident. This periodic property is common to the so-called voiced sounds that include the vowel sounds as well as many consonant sounds such as “n,” “l,” and “r.” Figure 12.14 gives an example of an image where, in

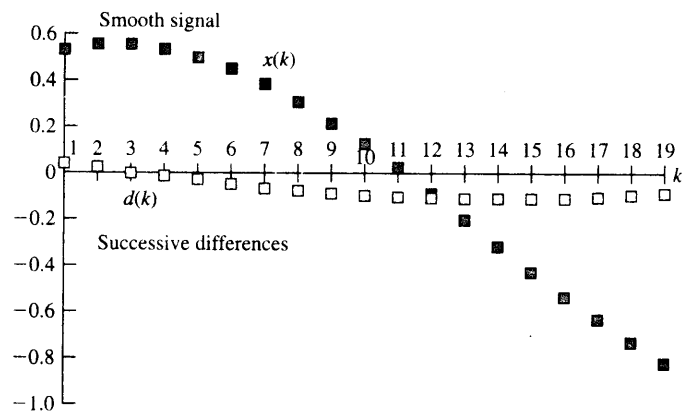


FIGURE 12.12 A smooth signal and its successive differences.



FIGURE 12.13 Sample waveform of "ae" sound as in cat.



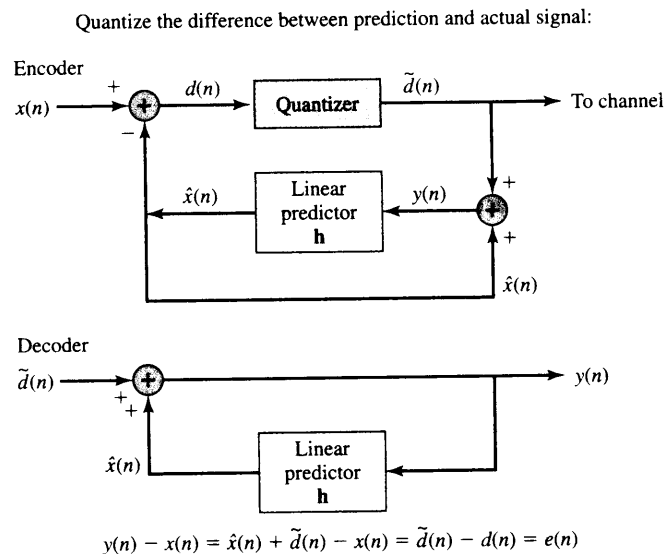
FIGURE 12.14 Image sample with smooth portions.

smooth portions of the picture, neighboring samples tend to have the same values. Long-term dependencies can also be observed when images contain periodic patterns. The compression techniques discussed in this section attempt to exploit the redundancies to attain greater compression and hence to provide more efficient representations.

Three basic compression techniques exploit the redundancies in signals. **Predictive coding** techniques, also called **differential coding** techniques, attempt to predict a sample value in terms of previous sample values. In Figure 12.12 the sequence of differences occupies a smaller dynamic range and consequently can be coded with greater accuracy by a quantizer. Predictive coding techniques are used extensively in the coding of speech signals. A second type of technique involves transforming the sequence of sample values into another domain that yields a signal that is more amenable to lossless data compression. These types of techniques are called transform coding techniques. Transform coding is used extensively in image and video coding. Subband coding is an important special case of transform coding. A third type of technique is to design quantizers that deal not with individual samples, but instead map blocks of samples into approximation points that have been designed to represent blocks of samples. This third technique, which is called **vector quantization**, is usually used in combination with other techniques. In the remainder of this section we discuss the first two types of techniques.

DIFFERENTIAL PCM

Figure 12.15 shows the block diagram of a **differential PCM (DPCM)** system, which is the simplest type of predictive coding. The next sample $x(n)$ is predicted by linear



The end-to-end error is only the error introduced by the quantizer!

FIGURE 12.15 Differential PCM coding.

combination of N previous outputs of the system:

$$\hat{x}(n) = h_1 y(n-1) + h_2 y(n-2) + \dots + h_N y(n-N) \quad (12.7)$$

In the simplest case we would use a first-order predictor where $\hat{x}(n)$ is predicted by $h_1 y(n-1)$. In general the prediction becomes more accurate as more terms are used in the predictor.

The difference between the sample value and the predicted value, which is called the **prediction error**, is applied to the quantizer, and the output is transmitted to the decoder. Note that the prediction is not in terms of the N previous samples $x(n-1)$, $x(n-2)$, \dots , $x(n-N)$, but rather in terms of the N previous outputs of the encoder. The reason is that the previous sample values are not available at the decoder. By using the above predictor, both the encoder and decoder will generate the same sequence of prediction values. It can be shown that the SNR of a DPCM system is given by

$$SNR_{DPCM} = SNR_{PCM} + 10 \log_{10} G_p \quad (12.8)$$

This equation has the following interesting interpretation. A PCM system in isolation has the SNR performance given by SNR_{PCM} . The introduction of the predictor reduces the power of the input to the quantizer from σ_x^2 to σ_d^2 . This reduction factor, given by $G_p = \sigma_x^2 / \sigma_d^2$, is directly translated into an improvement in SNR.

The design of a DPCM system involves finding the set of optimum prediction coefficients h_1, h_2, \dots, h_N that maximize the prediction gain. This process involves computing the statistics of a long sequence of speech samples from a variety of speakers and then carrying out certain matrix computations that ultimately yield the best predictor. See [Jayant and Noll 1984] for details. Typical values of G_p range from 6 to 10 dB. These gains can be used to reduce the bit rate by between one and two bits/sample. For example, the quality of a 64 kbps PCM, which uses eight bits/sample, can be obtained by using six bits/sample, giving a bit rate of six bits/sample \times 8000 samples/second = 48 kbps.

In the section on PCM it was indicated that the dynamic range of speech signals can vary considerably. The correlation properties of speech signals also vary with speaker and also according to the sound. The ITU has standardized a 32 kbps speech-coding method that enhances the preceding DPCM technique by using an adaptive quantizer as well as by computing the prediction coefficients on the fly in an attempt to maximize the instantaneous coding gain. This algorithm is called **adaptive DPCM (ADPCM)**. The encoded speech using this algorithm is of comparable subjective quality as that produced by eight-bit PCM coding. ADPCM is simple to implement and is used extensively in private network applications and in voice-mail systems.

Predictive coding techniques can be refined even further by analyzing speech samples so that the prediction coefficients are calculated essentially on a per sound basis. For example, for sounds such as the one in Figure 12.13 long-term predictors are used to predict the waveform from one period to the next. A further improvement involves replacing the coding of the prediction error with very low bit rate approximation functions. The class of **linear predictive coders (LPC)** uses this approach. Several variations of LPC coders have been selected for use in digital cellular telephone systems, for example, quadrature code-excited linear prediction (QCELP) coding at a bit rate of 14.4 kbps for use with the Qualcomm CDMA system, vector sum-excited linear prediction (VSELP)

coding at a bit rate of 8 kbps for use with TDMA, and regular pulse-excited long-term prediction (RPELTP) coding at a bit rate of 13 kbps for use in the GSM system.

In this section, we have focused on the application of predictive coding methods to speech signals. It should be noted, however, that the techniques are applicable to any signals where significant redundancy occurs between signal samples and are also used in image and video coding.

VOICE CODEC STANDARDS

A variety of voice codecs have been standardized for different target bit rates and implementation complexities. These include:

| | |
|---------|------------------------------|
| G.711 | 64 kbps using PCM |
| G.723.1 | 5–6 kbps using CELP |
| G.726 | 16–40 kbps using ADPCM |
| G.728 | 16 kbps using low delay CELP |
| G.729 | 8 kbps using CELP |

◆ SNR PERFORMANCE OF DPCM

We are interested in obtaining an expression for SNR performance of the DPCM system, which is given by the ratio of the average signal power and the average power of the end-to-end error introduced by the system. Let $d(n)$ be the prediction error

$$d(n) = x(n) - \hat{x}(n) \quad (12.9)$$

and let $\tilde{d}(n)$ be the output of the quantizer for the input $d(n)$. The end-to-end error of the system is given by $y(n) - x(n)$. By noting that $y(n) = \hat{x}(n) + \tilde{d}(n)$, we then have

$$y(n) - x(n) = \hat{x}(n) + \tilde{d}(n) - x(n) = \tilde{d}(n) - d(n) = e(n) \quad (12.10)$$

Equation 12.10 shows that the end-to-end error in the DPCM system is given solely by the error introduced by the quantizer. Thus the SNR of the system is given by

$$SNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{\sigma_x^2 \sigma_d^2}{\sigma_d^2 \sigma_e^2} = G_p \frac{\sigma_d^2}{\sigma_e^2} \quad (12.11)$$

where

$$G_p = \frac{\sigma_x^2}{\sigma_d^2} \quad (12.12)$$

and the first equality is the definition of the SNR and the second equality is obtained by multiplying and dividing by σ_d^2 . The term G_p is called the **prediction gain**, and the term σ_d^2/σ_e^2 is simply the performance of the quantizer in isolation because it is the ratio of the power of input to the quantizer and the power of the error introduced by the quantizer. In decibels the SNR equation (Equation 12.11) becomes

$$SNR_{DPCM} = 10 \log_{10} \frac{\sigma_d^2}{\sigma_e^2} + 10 \log_{10} G_p = SNR_{PCM} + 10 \log_{10} G_p \quad (12.13)$$

12.2.3 Transform and Subband Coding

Transform coding takes the sequence of signal sample values and converts them via a transformation into another sequence. In some instances the resulting sequence is more easily compressed or compacted. In other cases the transformed sequence allows the encoder to produce the quantization noise in a less perceivable form. Examples of the transformations that can be used are discrete Fourier and cosine transforms which involve taking time-domain signals into frequency-domain signals. A recently introduced example involves the use of wavelet transforms. In this and the next sections we consider subband coding and discrete cosine transform (DCT) coding and their application to speech, audio, image, and video signals.

When the number of quantization levels is not small, the quantization error signal has a flat spectrum. That is, its energy in the error signal is evenly distributed over the range of frequencies occupied by the signal. Most information signals have a spectrum that is nonuniform. Figure 12.16a shows the spectrum $X(f)$ of a typical signal and the associated quantization noise $Q(f)$. In this case the relative signal power to noise power differs according to frequency. In certain applications, such as audio, the ear is sensitive to these frequency-dependent SNRs. This situation suggests that systems could be designed to distribute the quantization error in a less perceivable form.

The subband coding technique was developed to do so. In **subband coding** the signal is decomposed into the sum of K component signals that are obtained by applying $x(t)$ to a bank of filters that are nonoverlapping in frequency. This procedure is shown in Figure 12.16b. Each component signal is quantized using a different quantizer. The number of levels in the quantizer determines the power of the quantization noise. Therefore, the encoder can control the amount of quantization noise introduced in each component through the number of bits assigned to the quantizer. The encoder typically performs the bit allocation based on the instantaneous power in the different subbands.

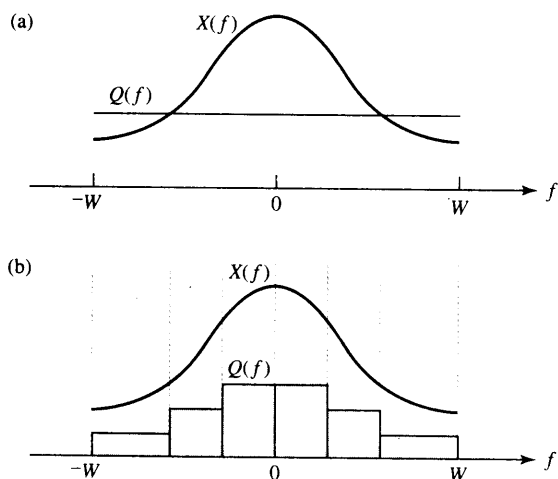


FIGURE 12.16 Subband coding of a typical signal. $X(f)$ is spectrum of original signal; $Q(f)$ is spectrum of error signal.

The MPEG standard for digital audio uses subband coding for the compression of audio.² Audio signals are sampled at a 16-bit resolution at sampling rates of 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, or 48 kHz. The sequence of time samples of the audio signal are filtered and divided into 32 disjoint component signals. The audio signal is also transformed into the frequency domain using a fast Fourier transform to assess the frequency composition of the signal. The subband signals are quantized according to the bit allocation specified by the encoder, which takes into account the instantaneous powers in the different frequency bands as well as their relative degree of perceptual importance. The **MPEG audio compression** standard has adjustable compression ratios that can deliver sound ranging from high-fidelity CD quality to telephone voice quality. MPEG audio compression can operate at three layers, from 1 to 3 in order of increasing complexity. A decoder of a given layer is required to decode compressed signals using all lower layers. The compressed signals have bit rates that range between 32 kbps to 384 kbps. The algorithm has a mode that allows it to compress stereo audio signals, taking into account the dependencies between the two channels.

WHAT IS MP3?

MP3 stands for MPEG layer 3 audio compression. The layer 3 compression option can reduce the bit rate of an audio signal by a factor of 12 with very low loss in sound quality. MP3 software for compression and playback can be downloaded from various websites. High-quality audio MP3 files can be created from music CDs, so MP3 has become very popular for “recording” selections from CDs in personal computers. The MP3 phenomenon has led to problems with the illegal distribution of copyrighted music material over the Internet. The recording industry is working on the development of secure digital recording methods.

12.3 IMAGE AND VIDEO CODING

Image and video information are multidimensional in nature and so differ from the speech and audio streams discussed in the previous section. Nevertheless, the techniques for compression can be generalized and extended to provide efficient representations of image and video signals. In this section we present the basic techniques that form the basis for current standards in image and video coding.

12.3.1 Discrete Cosine Transform Coding

Figure 12.17a shows samples of the signal that is smooth in the sense that it varies slowly in time. The figure also shows the corresponding **discrete cosine transform**

²MPEG video compression is discussed later in Section 12.3.4.

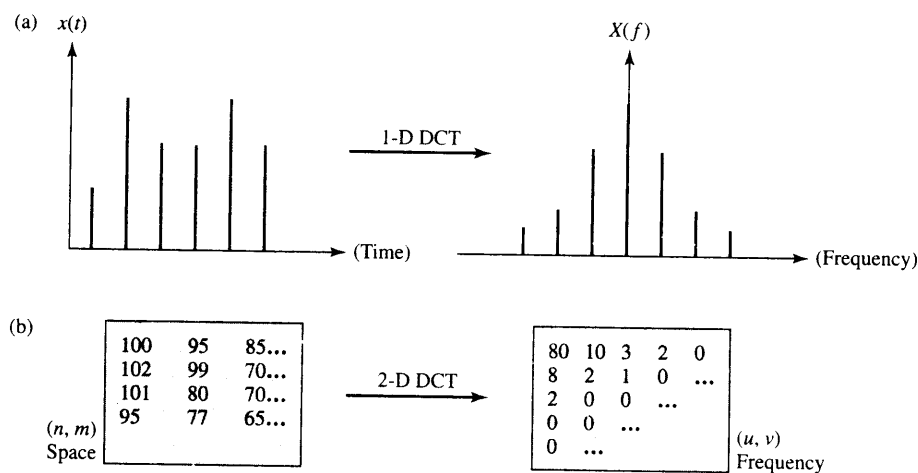


FIGURE 12.17 One-dimensional and two-dimensional DCTs.

(DCT) coding of the sequence. The smooth signal is primarily low-pass in nature so that the DCT of the signal is concentrated at low frequencies. If, instead of coding the signal in the time domain, we encode the signal in the frequency domain, we see that the frequency domain values will contain many zeros that can be encoded efficiently through the use of Huffman and run-length coding. In this section we refrain from the equations associated with the various transforms and focus on the qualitative behavior. Appropriate references are included at the end of the chapter.

The effectiveness of the DCT becomes more evident when the two-dimensional DCT is applied to images. In these applications the images are sampled to obtain a rectangular array of pixel values, as shown in Figure 12.18. This array is divided into square blocks consisting of 8×8 pixels. Figure 12.17b shows the values of the pixels obtained from a certain image. Note that in this case the time variable is replaced by the two space variables. The DCT then produces a block of values defined over two spatial frequency variables. The DCT of this 8×8 block is also shown in Figure 12.17b. Because the original block consisted of the smooth section of the image the DCT has nonzero values concentrated at the low frequencies that correspond to the upper-left corner of the transformed block. It can also be seen that all other pixel values are zero. It is clear that the transformed block can be more readily compressed.

12.3.2 The JPEG Image-Coding Standard

The **Joint Photograph Expert Group (JPEG)** image-coding standard uses the DCT in the following way. The original image is segmented into blocks of 8×8 pixel values. The DCT is then applied to each of these blocks. The coefficients in each DCT block are then quantized. Typically, a different quantizer is used for different frequency coordinates because of the differences in perceptual importance. The 8×8 array of pixels is converted into a block of 64 consecutive values through the zigzag scanning process shown in Figure 12.19. It can be seen that only the first few values in this

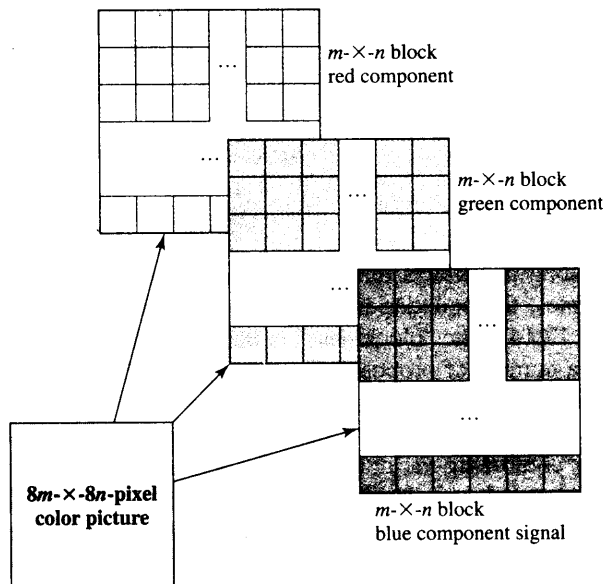


FIGURE 12.18 Scanning of an image: a color picture produces three color component signals.

block will be nonzero. The first component in each block, called the *dc component*, corresponds to the mean of the original pixel values and is encoded separately. The remainder of the components, the *ac components*, are encoded using Huffman codes designed to encode the nonzero values and run-length codes to deal with the long runs of zeros. Consecutive blocks tend to have similar means, so the dc components of consecutive blocks are encoded using DPCM. This technique can reduce the number of bits required to represent an image by compression ratios of 5 to 30. Figure 12.20 summarizes the JPEG image-coding standard.

The choice of 8 × 8-pixel blocks in the JPEG coding algorithm was primarily determined by the size of block for which the DCT algorithm could be implemented in VLSI. By transforming larger blocks of information, it is possible to capture the redundancies among larger groups of samples and hence achieve larger compression

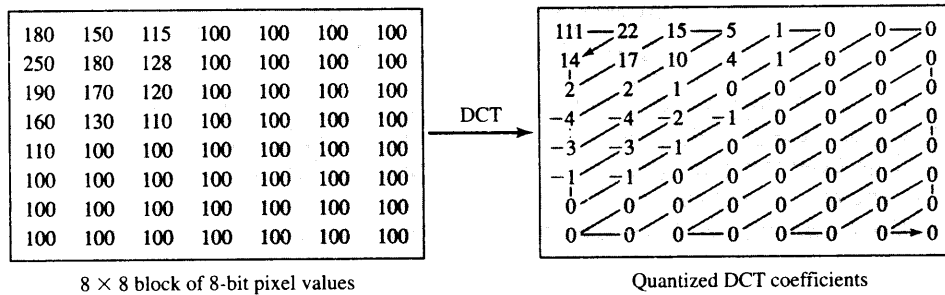


FIGURE 12.19 Zigzag scanning process: In image and video coding, the picture array is divided into 8 × 8 pixel blocks which are coded separately.

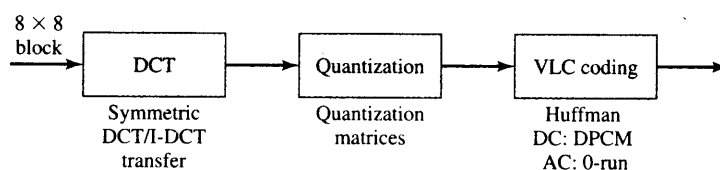


FIGURE 12.20 JPEG image-coding standard.

ratios. Another issue in JPEG coding is artifacts that result from the block-oriented coding. At very low coding bit rates, the boundaries between the blocks within the image become visible (see Figure 12.14). The “blockiness” in the picture can become quite annoying. There have been a number of proposals for using blocks that overlap so that adjacent blocks can be matched at the boundaries.

Consider now the question of dealing with color images. Modern video display systems produce color images by combining variable amounts of the three primary colors: red, green, and blue (RGB). Any color can be represented as a linear combination of these three colors. A color video camera or scanner produces an **RGB representation** which is a three-dimensional signal consisting of the three color components. Thus the compression of color images involves the compression of three component images, one for each primary color. For example, “full color” display systems use eight bits to represent each color component, giving a total of 24 bits/pixel and a capability of representing $2^{24} = 16,777,216$ possible colors. An SVGA system has a resolution of 800×600 pixels/screen, and so it requires 1.44 megabytes of memory to store one screen. Similarly, an XGA system has a resolution of 1024×768 pixels/screen, and so it requires $1024 \times 768 \times 3 = 2.4$ megabytes of memory. If you play with the resolution control in your personal computer, you will be given a number of choices for the resolution and the total number of colors that the available memory can accommodate.

To compress a color image, the above DCT coding algorithm can be applied separately to each component image. A more effective compression method involves first applying a transformation to the RGB representation of the image. The R, G, and B components for a pixel are converted into a luminance component, Y , and two chrominance components, I and Q , by the following equations:

$$x_Y = 0.30x_R + 0.59x_G + 0.11x_B \quad (12.14)$$

$$x_I = 0.60x_R - 0.28x_G - 0.32x_B \quad (12.15)$$

$$x_Q = 0.21x_R - 0.52x_G + 0.31x_B \quad (12.16)$$

This transformation was developed as part of the design of color television systems that required compatibility with black-and-white television. The luminance component provides the information required to produce the image in a black-and-white television. The two chrominance components provide the additional information required to produce a color image. One of the observations made in the design of color television was that the human eye is much more sensitive to the luminance signal than to the chrominance signals because the luminance signal provides information about edges and transitions in images. An analogy can be made to the coloring books that you surely grew up with; the black lines on a page that define the basic image correspond

to the luminance signal, and the coloring on the page corresponds to the chrominance components. Image compression systems make use of this difference in sensitivity by processing the luminance signal at the original resolution and processing the chrominance signals at lower resolutions.

A commonly used format involves having the chrominance components represented at one-quarter of the resolution of the luminance signal. For example, one frame of a television signal consists of 480 lines with 640 pixels/line. This frame will produce 4800 8×8 -pixel blocks for the luminance signal for one frame. The chrominance signals are originally sampled at this resolution but blocks of 2×2 pixels are collapsed into a single pixel by taking an average of the four pixel values. As a result, each chrominance signal will produce 1200 pixel blocks for each frame. The DCT compression algorithm can then be applied to the blocks produced by the three component signals and the resulting representation can be transmitted or stored. Note that this approach halves the total number of blocks that need to be processed from $3 \times 4800 = 14,400$ blocks to $4800 + 2 \times 1200 = 7200$ blocks. The decoder then recovers the luminance frame and the two chrominance frames. The latter are expanded to the original resolution by creating a 2×2 set of pixels from each pixel value.

As an example, consider a scanned image of a $8'' \times 10''$ color picture at a resolution of 1200 pixels/inch. Inexpensive scanners with this resolution are now available. The number of pixels produced in the scanning process is then

$$8 \times 1200 \times 10 \times 1200 = 115.2 \times 10^6 \text{ pixels} \quad (12.17)$$

Assuming that 24 bits are used to represent each pixel value, the uncompressed scanned picture will yield a file consisting of 350 megabytes! If the JPEG algorithm can produce a high-quality reproduction with a compression ratio of 10 for this particular image, then the file size is reduced to 35 megabytes. If in addition we use one-quarter resolution for the chrominance components, the file size can then be reduced to about 17.5 megabytes.

12.3.3 Compression of Video Signals

Television and motion pictures are based on the principle that a rapid sequence of pictures can give the appearance of motion. Consider, for example, a booklet of pictures that change incrementally. When the pages of the booklet are flicked in rapid succession, the sequence of pictures merge to produce animation. This principle is the basis for motion pictures where a series of photographs are projected onto a screen at a rate of 24 picture frames/second. Figure 12.21 shows how the principle is incorporated into television.

The television signal consists of a one-dimensional function of time that is used to produce a sequence of images. The television signal specifies the intensity that is to be drawn in a horizontal line across the television screen. **Synchronization** signals control the tracing of consecutive lines down the screen as well as the return to the top of the screen after each picture frame. The television system in North America and Japan uses 525 lines/frame and displays 30 frames/second. Systems in Europe used 625 lines/frame and a display rate of 25 frames/second. At these rates some flicker is still noticeable, but instead of increasing the frame rate, a technique called **interlacing** is used. A frame is

Information = M bits/pixel \times ($W \times H$) pixels/frame $\times F$ frames/second

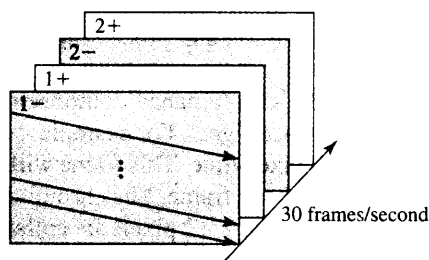


FIGURE 12.21 The TV/video signal.

divided into two fields that consist of the odd and even lines, respectively. Odd (–) and even (+) fields are displayed in alternation, resulting in a display rate of 60 fields/second. The odd and even fields draw lines in nonoverlapping portions of the picture. The display screen consists of material that maintains each line for a sufficient period of time to produce the appearance of a higher frame rate. Modern technology now makes it possible to implement progressive or noninterlaced video systems that use higher frame rates.

From Figure 12.21 we can calculate the uncompressed bit error rate required to represent a standard television signal. In the figure we assume that each frame consists of 720 lines by 480 pixels/line, as specified by an industry standard for digital television. If we use eight bits to represent each color component of a given pixel, then the bit rate for the uncompressed television signal is $24 \times 720 \times 480 \times 30 = 248$ megabits/second. As another example, consider one of the high-definition television standards that has a resolution of 1920×1080 pixels/frame at 60 frames/second. The uncompressed bit rate for this HDTV signal is 3 gigabits/second. Clearly, effective compression algorithms are desirable for these signals.

In the discussion relating to JPEG, we presented a DCT coding method as a means for exploiting the spatial redundancies in a picture. In **intraframe video coding** each picture is encoded independently of all other pictures. The motion JPEG system involves applying the JPEG algorithm to each frame of a video sequence. For the 720×480 television signal described above, the JPEG algorithm can produce digital video with high quality at bit rates of about 20 megabits/second.

The temporal nature of video implies that video signals also contain temporal redundancy. **Interframe video coding** methods exploit this redundancy to achieve higher compression ratios. Video signals can be viewed as consisting of scenes in which the setting is fixed and the time is continuous. In scenes with little or no motion, the consecutive frames differ in very small ways. In this case the notion of predictive coding suggests that we encode the differences between consecutive frames. The pixels in these different frames will predominantly consist of zeros and cover a narrow dynamic range. From our results on the performance of quantizers, we know that such signals will be highly compressible. However, in scenes that contain a significant amount of motion the difference between successive frames will fail to capture the temporal relation between them. Motion compensation algorithms have been developed to address this problem.

Figure 12.22 shows the operation of the video compensation method. As before, frames are segmented into 8×8 -pixel blocks. The coding of given frame is performed

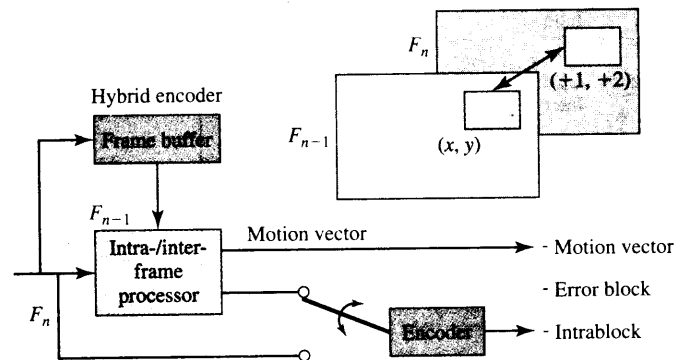


FIGURE 12.22 Video motion compensation.

relative to the previous frame, which is stored in a buffer. For each 8×8 block of pixels, a search is carried out to identify the 8×8 block in the preceding frame that best matches the block that is to be encoded. A **motion vector** is calculated specifying the relative location of the encoded block to the block in the preceding frame. The DCT is applied to the difference between the two blocks, and the result is quantized and encoded as in the JPEG method. The resulting compressed information and motion vector allow the decoder to reconstruct the approximation of the original signal.

Motion compensation is the most time-intensive part in current video compression algorithms. The region that is searched to identify the best fit to a block is typically limited to a region in the vicinity of the block. Note that the algorithm that is used to conduct the search does not need to be standardized, since the decoder needs to know the motion vectors but not how they were found. Another consideration in selecting the motion compensation algorithm is whether the encoding has to be done in real time. In a studio production situation where the uncompressed video sequence has been prerecorded and stored, the encoder can use complex motion compensation algorithms to produce efficient and high-quality compressed versions of the signal. In real-time applications, such as broadcasting and videoconferencing, the system has much less time to carry out motion compensation, so simpler, less effective algorithms are used.

Figure 12.23 shows the block diagram for the ITU H.261 video encoding standard for videoconferencing applications. This standard provides for various coding options that can result in bit rates of the form $p \times 64$ kbps. Recall that 64 kbps is the standard bit rate required to handle a single voice signal within the telephone network. The $p = 1$ and $p = 2$ versions can be accommodated in the basic rate interface of ISDN. The $p = 24$ version occupies a T-1 telephone line. The H.261 system is defined for frames of size 360×240 pixels and 180×120 pixels, which are $1/4$ and $1/16$ the resolution of the 720×480 system discussed earlier. The encoder uses a combination of intraframe DCT coding and interframe DCT coding with motion compensation. Reasonable picture quality is provided by these systems when the scenes contain little motion. However, the encoding delay is significant, so the time that elapses from when a person moves to when the motion is seen on the screen is noticeable. This delay makes the interaction between the videoconferencing participants a bit awkward.

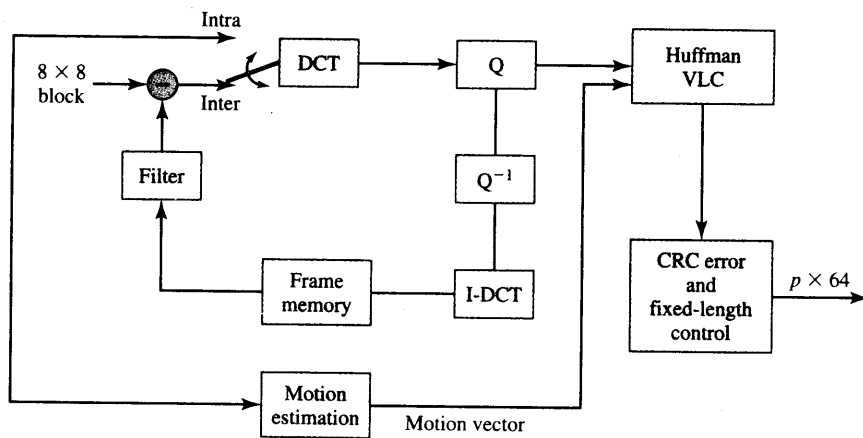


FIGURE 12.23 The H.261 videoconferencing standard.

12.3.4 The MPEG Video Coding Standards

Video on demand and related applications require that the encoded video signal accommodate capabilities associated with VCR controls. These include the ability to fast forward, reverse, and access a specific frame in a video sequence. The *Motion Picture Expert Group (MPEG)* standards have been developed to meet these needs. The **MPEG-1** and **MPEG-2 standards** use three types of encoding that produce three types of frames as shown in Figure 12.24. **I-frames** are encoded with intraframe coding. Since their decoding is independent of all other frames, I-frames facilitate fast-forward, reverse, and random-access capabilities. I-frames have the lowest compression, but they

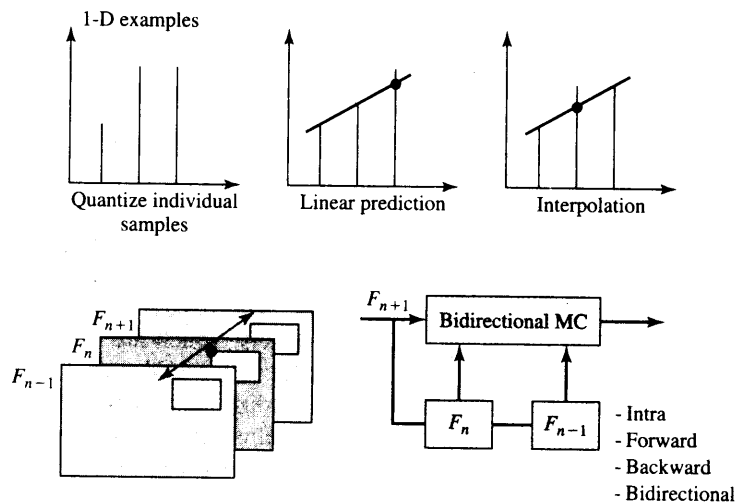


FIGURE 12.24 Prediction and interpolation in MPEG.

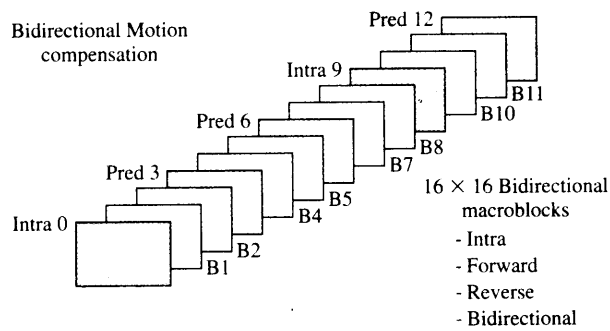


FIGURE 12.25 MPEG group of picture structure.

can be encoded and decoded faster than the other frame types. **P-frames** (predictive) are encoded by using motion compensation relative to the most recent I- or P-frame. P-frames have better compression levels than I-frames. **B-frames** (bidirectional) use motion compensation relative to both the preceding and the following I- or P-frames. The motion vector for a given block can refer to either or both of these two frames. B-frames achieve the highest compression of the three frame types, but they also take the longest time to encode. In addition, they cause significant delay at the encoder, since they require the availability of the following I- or P-frame. In the MPEG standards the motion compensation algorithm attempts to find the best match for macroblocks that consist of 16×16 -pixel blocks of luminance symbols. The motion vector that results from this process is then used for the associated 8×8 -pixel chrominance signal blocks.

The encoded frames in MPEG are arranged in **groups of pictures** as shown in Figure 12.25. I-frames are inserted into the sequence at regular intervals to provide VCR capabilities as well as to limit the propagation of errors associated with predictive coding. The remainder of the frames consist of P-frames and B-frames.

The MPEG-1 standard was developed to produce VCR-quality digital video at about 1.2 Mbps. This standard makes storage possible in CD-ROM and transmission possible over T-1 digital telephone lines. The MPEG-2 standard is much broader in scope. It can accommodate frame sizes of 352×240 pixels, 720×480 pixels, 1440×1152 pixels, and 1920×1080 pixels. The MPEG-2 standard defines a system of profiles where each consists of a set of coding tools and parameters. For example, the MPEG-2 main profile uses three types of frames, namely, I-, B-, and P-frames. It focuses on the 720×480 -pixel resolution that corresponds to conventional television. Very good quality video is attainable using MPEG-2 at bit rates in the range of 4 to 6 Mbps. Another profile accommodates simpler implementation by avoiding the use of B-frames.

Other profiles address scalability options where users with differing requirements wish to access the same video information, as shown in Figure 12.26. In **SNR scalability** different users require the same image sizes but have different bit rates available. The encoded video then needs to be represented as two streams: one at a basic quality that meets the lower of the bit-rate requirements; the other that provides information that allows the higher bit-rate user to enhance the quality of the recovered signal. SNR scalability is also referred to as layered coding because it can provide the basis for maintaining basic-quality transmission in the presence of errors. This level of performance is achieved by having the base stream transmitted with higher priority or

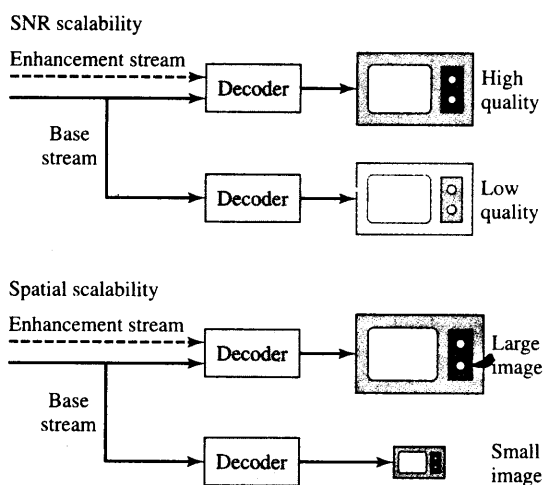


FIGURE 12.26 Scalable video coding.

greater error protection than the enhancement stream. In **spatial scalability** different users wish to access the same information at different resolutions. Again, the encoded information is divided into two streams: one that provides the basic resolution and the other that provides the enhancement required to obtain the higher resolution. The aim of spatial scalability in MPEG-2 is to support multiresolution applications that involve backward compatibility, for example, between MPEG-2 and MPEG-1 or H.261, as well as compatibility between standard-format television and HDTV.

The MPEG-2 profiles also consider HDTV. As discussed in Chapter 3, conventional television uses an aspect ratio of 4:3, giving a squarish picture. HDTV uses an aspect ratio of 16:9, giving a picture that is closer to that of motion pictures. MPEG-2 allows for two larger resolutions to accommodate HDTV: 1440×1152 and 1920×1080 . The higher resolution and frame rates of these signals result in huge bit rates for the uncompressed signal. MPEG-2 coding can produce high-quality HDTV compressed signals at bit rates in the range 19 to 38 Mbps. The Advanced Television System Committee (ATSC) has developed an MPEG-2-based coding standard for HDTV systems in the United States for use over terrestrial broadcast and cable systems over conventional analog 6 MHz bandwidth channels.

The MPEG standards group has already begun work on an MPEG-4 coding standard for use in *very low bit rate* applications such as those encountered in wireless networks. It is clear that the requirement for multimedia communications will soon extend to wireless networks. The problems are that bandwidth is limited and communication is unreliable in wireless networks. So the challenge here is how to send images, for example, remote telemetry or security, over these types of channels.

12.3.5 MPEG Multiplexing

The systems part of the MPEG-2 standard deals with the multiplexing and demultiplexing of streams of audio, video, and control information. MPEG-2 has the ability to

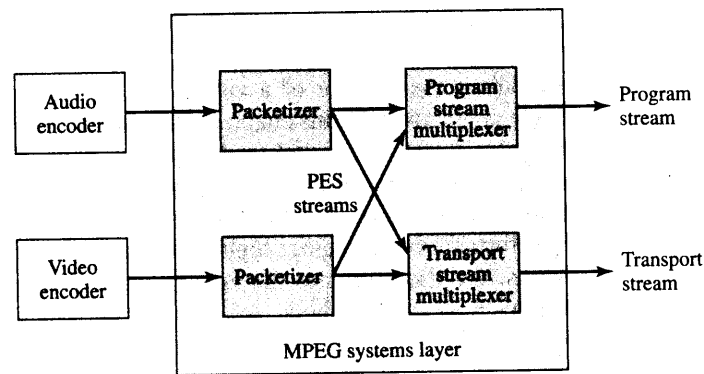


FIGURE 12.27 MPEG-2 multiplexing.

carry multiple elementary streams that need not have a common time base. Figure 12.27 shows the case of an audio and a video stream that have a common time base. The outputs of the encoders are packetized, and timestamps derived from a common clock are inserted into each packet. This process results in *packetized elementary streams (PES)*. Each PES packet contains information that identifies the stream; specifies the packet length, and gives a PES priority that can be used with layered coding, as well as presentation and decoding timestamps and packet transmission rate information. As shown in the figure, two multiplexing options are supported: program stream and transport stream.

MPEG-2 program stream results from the multiplexing of PES streams that have a common time base into a single stream. The program stream is intended for error-free environments, and so it uses packets that are variable length and relatively long. The transport stream multiplexes one or more PES streams with one or more time bases. The transport stream is intended for situations where packet loss may occur, for example, network transmission, and so it uses fixed-length 188-byte packets that facilitate recovery from the losses. The transport stream packets contain program clock reference timestamps that can be used for clock recovery.

For a transport stream the decoder demultiplexes the stream and removes the transport stream and PES headers to reconstruct the compressed video and audio elementary streams. These streams are placed in a buffer and retrieved by the video and audio decoders at the appropriate time. Two timing processes are in play here. First the output from the video and audio decoders must synchronize to have “lip synch.” This process requires the synchronization of the two media streams and hence the compensation for any misalignments in the relative timing of packets that may have occurred during transmission. The synchronization is done through the use of presentation timestamps and decoding timestamps. The second timing process is the recovery of the encoder clock. The decoder must reconstruct the timing of the clock that was used at the encoder. The encoder inserts program clock reference timestamps. Chapter 5 covers this type of timing recovery.

SUMMARY

Multimedia networking involves the transfer of a variety of information types that may be integrated in a single application. In this chapter we presented the compression techniques that are used to obtain efficient digital representations for various types of information media. We took a close look at the format, bit rates, and other properties of different types of information, such as speech, audio, facsimile, image, and video. We saw that audio and video information can require higher bit rates than those typically provided by packet networks. We also presented the important compression standards from the classic logarithmic-PCM for telephone voice to the current MPEG-2 and MP3 standards for video and audio.

CHECKLIST OF IMPORTANT TERMS

| | |
|---|--------------------------------|
| adaptive lossless data compression | motion vector |
| adaptive DPCM (ADPCM) | MPEG audio compression |
| adaptive quantizer | MPEG-1, MPEG-2 coding standard |
| B-frame | nonuniform quantizers |
| clipping | P-frame |
| codeword | pixel |
| differential coding | prediction error |
| differential PCM (DPCM) | prediction gain |
| discrete cosine transform (DCT) | predictive coding |
| ♦ entropy | pulse code modulation (PCM) |
| groups of pictures | RGB representation |
| Huffman code | run |
| I-frame | run-length code |
| interframe video coding | SNR scalability |
| interlacing | spatial scalability |
| intraframe video coding | subband coding |
| Joint Photograph Expert Group (JPEG) | synchronization |
| linear predictive coders (LPC) | transform coding |
| | vector quantization |

FURTHER READING



- Gibson, J. D., T. Berger, T. Lookabaugh, D. Lindbergh, and R. L. Baker, *Digital Compression for Multimedia: Principles and Standards*, Morgan Kaufmann, San Francisco, 1998
- Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice Hall, Englewood Cliffs, New Jersey, 1984.
- Seuss, Dr. [pseudo.], *Hop on Pop*, Random House, New York, 1963.
- See our website for additional references available through the Internet.*

PROBLEMS

- 12.1. (a) Give three examples of applications in which the information must be represented in a lossless way.
 (b) Give three examples of applications in which information can be represented in a lossy manner.
- 12.2. The probabilities for the letters in the English alphabet are given in the table below. The space between letters constitutes one-sixth of the characters in a page.
 (a) Design a Huffman code for the letters of the English alphabet and the space character.
 (b) What is the average number of bits/symbol?
 (c) Compare the answer of part (b) to the entropy.

| | Probability of occurrence | | Probability of occurrence | | Probability of occurrence | | Probability of occurrence |
|---|------------------------------|---|------------------------------|---|------------------------------|---|------------------------------|
| A | 0.08149 | H | 0.05257 | O | 0.07993 | U | 0.02458 |
| B | 0.01439 | I | 0.06344 | P | 0.01981 | V | 0.00919 |
| C | 0.02757 | J | 0.00132 | Q | 0.00121 | W | 0.01538 |
| D | 0.03787 | K | 0.00420 | R | 0.06880 | X | 0.00166 |
| E | 0.13101 | L | 0.03388 | S | 0.06099 | Y | 0.01982 |
| F | 0.02923 | M | 0.02535 | T | 0.10465 | Z | 0.00077 |
| G | 0.01993 | N | 0.07096 | | | | |

- 12.3. Suppose an information source generates symbols from the alphabet {a, b, c} and suppose that the three symbols are equiprobable.
 (a) Find the Huffman code for the case where the information is encoded one symbol at a time?
 (b) Find the Huffman codes when the symbols are encoded in blocks of length 2 and in blocks of length 3.
 (c) Compare the answers in parts (a) and (b) with the entropy of the information source.
- 12.4. Consider a binary information source that “stutters” in that even-numbered symbols are repetitions of the previous odd-numbered symbols. The odd-numbered symbols are independent of other symbols and take on the values 0 and 1 with equal probabilities.
 (a) Design a Huffman code for encoding pairs of symbols where the first component of the pair is an odd-numbered symbol and the second component is the following even-numbered symbol.
 (b) Now design a Huffman code for encoding pairs of symbols where the first component of the pair is an even-numbered symbol and the second component is the following odd-numbered symbol.
 (c) Compare the performance of the two codes. What is the entropy of this information source?
- 12.5. The MNPS lossless data compression algorithm used in various modem products uses a combination of run-length coding and adaptive variable-length coding. The algorithm is designed to encode binary octets, that is, groups of eight bits. The algorithm keeps a running count of the frequency of occurrence of the 256 possible octets and has a table listing the octets from the most frequent to the least frequent. The two most frequent

octets are given the codewords 0000 and 0001. The remaining 254 octets are assigned a codeword that consists of a three-bit prefix that specifies the number of remaining bits in the codeword and a suffix that specifies the rank of the codeword within a group:

| | | |
|-----------------------------|-----|---------|
| most frequent octet | 000 | 0 |
| second most frequent octet | 000 | 1 |
| third most frequent octet | 001 | 0 |
| fourth most frequent octet | 001 | 1 |
| fifth most frequent octet | 010 | 00 |
| sixth most frequent octet | 010 | 01 |
| seventh most frequent octet | 010 | 10 |
| eighth most frequent octet | 010 | 11 |
| ninth most frequent octet | 011 | 000 |
| ... | | |
| 16th most frequent octet | 011 | 111 |
| 17th most frequent octet | 100 | 0000 |
| ... | | |
| 32nd most frequent octet | 100 | 1111 |
| ... | | |
| 129th most frequent octet | 111 | 0000000 |
| ... | | |
| 256th most frequent octet | 111 | 1111111 |

- (a) Suppose that a certain source generates the 256 octets with respective probabilities $c(1/2)^i$, $i = 1, 2, \dots, 256$, where $c \approx 1$. What is the average number of bits/octet? What is the entropy of this source?
- (b) Suppose that the 256 octets are equiprobable? What is the average number of bits/octet? Is coding useful in this case?
- (c) For which set of probabilities is this the Huffman code? What is the entropy for such a source?
- 12.6.** (a) Use the $m = 3$ and $m = 5$ run-length codes to encode the binary string used in Figure 12.5.
- (b) Repeat part (a) using the $m = 3$ and $m = 5$ codes from Figure 12.6.
- 12.7.** Consider a binary source that produces information bits b_i independent of each other and where $P[b_i = 0] = p$ and $P[b_i = 1] = 1 - p$. Let l be the number of consecutive 0s before a 1 occurs.
- (a) Show that $P[l = i] = (1 - p)p^i$, for $i = 0, 1, 2, \dots$
- (b) Show that $P[l > L] = p^{L+1}$.
- (c) Find the performance of the run-length code in Figure 12.5 for $m = 4$ when $p^{16} = 1/2$.
- (d) Find the performance of the run-length code in Figure 12.7 for $m = 4$ when $p^{16} = 1/2$ and compare it to the result in part (c).
- 12.8.** (a) Use the Lempel-Ziv algorithm, as described in the chapter, to encode the string of characters in part (c) and part (d) of Problem 12.7.
- (b) Repeat (a) where the algorithm refers to the most recent occurrence of the repeated pattern.

- 12.9.** Decode the following Lempel-Ziv encoded passage from Dr. Seuss and find the compression ratio:

```
I_do_not_like_them_in_a_box._[1,19]with[22,3]f[26,4][1,24]house
[28,28]m[86,25][16,2]r[13,2]or[14,2][144,4][123,2][1,19]anyw[153,20]
gree[81,2]eggs[177,3]d[84,2]am[28,21]s[218,2]-I-[218,3]
```

- 12.10.** Apply the Lempel-Ziv algorithm to the binary input string in Figure 12.5.
- 12.11.** What are the consequences of transmission errors in PCM? What are the consequences in DPCM?
- 12.12.** (a) Use the features in the sample waveforms shown in Figures 12.11 and 12.13 to explain how the ADPCM algorithm achieves compression.
 (b) What additional information, relative to DPCM, does the ADPCM encoder have to transmit to the decoder?
- 12.13.** Suppose that a sound such as that shown in Figure 12.13 has a duration of 30 ms. Suppose that you compute a Fourier series to produce a periodic function to approximate the signal.
 (a) Estimate the number of bits required to specify the Fourier series.
 (b) Now suppose you take the difference between the periodic function produced by the Fourier series and the actual function. How would you encode the difference signal to produce an improved quality signal? Give a rough estimate of the additional number of bits required?
- 12.14.** If you have a microphone and a utility such as Sound Blaster Wave Studio in your PC that allows you to sample speech waveforms, describe the properties of the following sounds: “e” (as in *beet*), “i” (as in *bit*), “m” (as in *moon*), “s” (as in *sit*), and “h” (as in *hit*). How do the first three waveforms differ from the latter two?
- 12.15.** In Figure 12.16a is the subband coding system better off not transmitting the outer band where the noise level exceeds the signal level?
- 12.16.** Current AM transmission signals occupy a band of 10 kHz. Suppose that we wish to transmit stereo CD-quality audio over an AM frequency band.
 (a) What is the minimum size of a modem signal constellation required to achieve this using uniform quantization? Refer to QAM in Chapter 3.
 (b) What is the minimum size of signal constellation if MP3 is used instead?
- 12.17.** FM radio signals use a transmission bandwidth of 200 kHz. Can MPEG-1 or MPEG-2 video be transmitted over one of these channels? Explain.
- 12.18.** An analog TV channel occupies a bandwidth of 6 MHz. How many MPEG-1 channels can be carried over such a channel if a digital transmission scheme is used with four constellation points?
- 12.19.** (*The Globe and Mail*, Jan. 25, 1995) The CD-video standard proposed by Sony/Phillips provides for movies of up to 135 minutes with a total storage of 3.7 Gbytes. Can the JPEG or MPEG coding standards meet these bit rate requirements?

- 12.20.** How many minutes of music can be stored in the CD system of Problem 12.19, using uniform quantization? MP3 coding?
- 12.21.** An XGA graphics display has a resolution of 1024×768 pixels. How many bytes are required to store a screenful of data if the number of colors is 256? 65,536? 16,777,216?
- 12.22.** A scanner can handle color images up to a maximum size of 8.5 inches by 11 inches at resolutions up to 4800 pixels/inch. What file size is generated by the maximum-size image sampled at 24 bits/pixel? What range of file size will result if JPEG compression is used?
- 12.23.** In JPEG the DCT coefficients are expressed as numbers in the range 0 to 255: The DCT coefficients (prior to quantization) corresponding to an 8×8 block of pixels is given below:

| | | | | | | | |
|-----|----|----|----|---|---|---|---|
| 148 | 92 | 54 | 20 | 6 | 2 | 2 | 0 |
| 86 | 72 | 45 | 16 | 8 | 1 | 0 | 0 |
| 56 | 48 | 32 | 10 | 7 | 2 | 0 | 0 |
| 20 | 14 | 8 | 4 | 2 | 0 | 0 | 0 |
| 4 | 3 | 2 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In JPEG a DCT coefficient is quantized by dividing it by a weight 2^k and rounding the quotient, where $k = 0, 1, \dots, 6$ is an integer selected according to the relative importance of the DCT coefficient. Coefficients corresponding to the lower frequencies have small values of k , and those corresponding to higher frequencies have higher values of k . The 8×8 quantization table specifies the weights that are to be used.

- (a) Explain how the larger values of k correspond to coarser quantizers.
 (b) From the following matrix of quantization table, find the resulting block of quantized DCT coefficients.

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 1 | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 1 | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 2 | 2 | 2 | 4 | 8 | 16 | 32 | 64 |
| 4 | 4 | 4 | 4 | 8 | 16 | 32 | 64 |
| 8 | 8 | 8 | 8 | 8 | 16 | 32 | 64 |
| 16 | 16 | 16 | 16 | 16 | 16 | 32 | 64 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 64 |
| 64 | 64 | 64 | 64 | 64 | 64 | 64 | 64 |

- (c) Find the one-dimensional sequence that results after zigzag scanning.
- 12.24.** (a) What impact does the speed of a CD-ROM drive have on the applications it can support?
 (b) How does speed interact with total storage capacity?
- 12.25.** A DS-3 digital transmission system has a bit rate of 45 Mbps and is the first level of high-speed transmission available to users.
 (a) How many PCM calls can be accommodated in one DS-3 line?
 (b) How many MPEG-1 or MPEG-2 television channels can be similarly accommodated?

- 12.26.** What minimum delay does the MPEG decoder have to introduce to decode B frames? What impact does this factor have on the storage required at the receiver?
- 12.27.** Suppose a video signal is to be broadcast over a network to a larger number of receivers and that the receivers have displays with resolutions of 720×480 , 360×240 , and 180×120 . Explain how the scalability options of MPEG coding may be used in this multicasting application.
- 12.28.** A video server system is designed to handle 100 simultaneous video streams. Explain the tasks that need to be carried out in servicing one stream. What is the aggregate input/output requirements of the system?

Trends in Network Architectures

We have reached the end of our journey only to find that the story is not all told. Networks are evolving at a rapid pace, and many changes are yet to come. We hope we have succeeded at least partially in teaching the student the fundamentals that are lasting and that provide the means to read the future in the present. We end with a brief discussion of current trends in network architecture.

In Chapter 1 we noted how network architecture is fundamentally influenced by the cost of bandwidth and the cost of processing, both of which have been steadily declining. Let us examine the impact of these costs on network architecture.

The decrease in the cost of electronic processing has resulted in greater protocol processing capability. Faster QoS-capable switches and routers result from hardware-based packet classifiers, prefix matching, schedulers, and interconnection packet-forwarding fabrics. Improved speed and capability in software-based functions also result from lower cost and faster microprocessors. Notwithstanding these advances, the scalability problems inherent in networking continue to pose a challenge with the tremendous rate of growth in the Internet.

The cost of bandwidth in the form of optical transmission has also been steadily decreasing, but with the introduction of dense wavelength-division multiplexing (DWDM) systems a major discontinuity will occur. A single fiber can now carry 160 wavelengths at 10 Gbps/wavelength. When one considers that a duct can contain several hundred optical fibers, it is clear that routes carrying petabits/second (10^{15} bps) will soon become available. This abundant available bandwidth will eventually have a dramatic effect on the architecture of the core network, surely in a way that addresses existing scalability challenges.

DWDM with associated wavelength-selective cross-connects provide a circuit-based capability for creating virtual networks that can be reconfigured to meet the demand at any given point in time. This DWDM infrastructure provides network-protocol flexibility in that the digital stream carried in each wavelength may be agnostic to the bit rate and format of the actual stream. The overall network capacity will be

increased further with the availability of wavelength interchangers that can switch an incoming wavelength into a different outgoing wavelength; this technology will allow a greater degree of wavelength reuse.

Routers with terabit/second capacity are already under development to connect to the optical pipes that will become available with DWDM. Router companies are already claiming designs that can handle links at speeds of 10 Gbps, and even 40 Gbps. Efforts are also under way to standardize "light" framing procedures for encapsulating IP packets directly onto optical links. Researchers have been trying to develop all-optical packet switching, however, these efforts have not been successful. Optical processing and buffering capabilities continue to lag electronic counterparts in terms of density, cost, power, and reliability. We predict that true all-optical packet switching remains many years into the future.

Traffic from an array of access networks will be aggregated in community and metropolitan networks and fed into the terabit routers and switches that connect to the core network. Access networks will be quite diverse, ranging from conventional telephone and LAN access to newer forms of access such as DSL and cable modem. Despite the growth of IP traffic, telephone traffic remains a very significant source of revenue and so the telephone network infrastructure is not about to disappear. A new generation of SONET electronic multiplexing and switching equipment is being deployed that provides the reliability and QoS required by voice services but that also accommodates the requirements of data services. Consequently, the next generation of multiservice networks will consist of a mix of optical, SONET, and Ethernet/IP gear.

An interesting and unexpected development in the industry has been the emergence of generalized MPLS (GMPLS) as a means for controlling and managing multiservice networks. GMPLS extends the signaling capabilities of MPLS to enable the establishment of generalized paths across multiservice networks. These paths can be label-switched packet paths, SONET connections, or optical wavelength paths (often called light paths). The use of a common signaling framework for all connection types is very powerful. GMPLS also extends the resource management capabilities of OSPF for multiservice networks. Optical, SONET, and packet resources (switches, multiplexers, links) can now be tracked, monitored, and provisioned automatically and in an integrated manner. This approach promises to result in huge savings in the cost of operating networks, a critical need for the telecom industry that is currently in dire straits.

In another direction, new research is aimed at reducing the capital and operating cost of a network by collapsing network layering in a multiservice network into one optical layer. The overall architecture is made up of a network core consisting of wavelength-selective cross-connects and a network edge consisting of fast tunable lasers and burst-mode receivers. Complex operations in the network core are removed by emulating fast switching in the core through fast tuning of lasers at the edge. This approach essentially takes advantage of wavelength division multiplexing in the core capable of supporting high bandwidth, and time interleaving at the edge capable of multiplexing traffic of fine bandwidth granularities.

Various new types of wireless access networks are emerging: in-building LANs using the 802.11 wireless standards have mushroomed, a new generation of cellular networks capable of supporting voice and data is under deployment, various fixed wireless access networking technologies have been proposed and are being tested, and the possibility of all-IP wireless networks is becoming real.

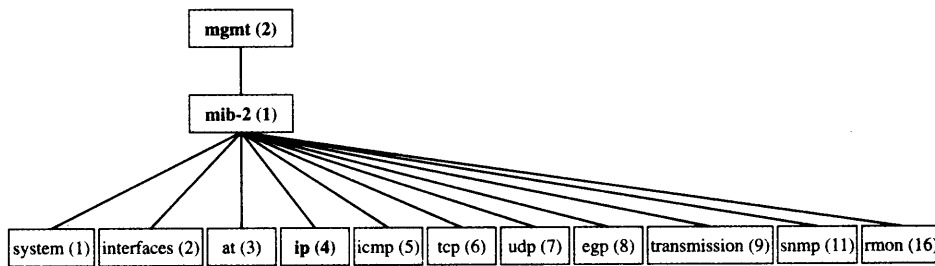


FIGURE B.4 Standard system objects of mib-2 (1) subtree.

- The security (5) subtree is for objects related to security.
- The snmpv2 (6) subtree is reserved for housekeeping purposes for SNMPv2. This subtree includes object information for transport domains, transport proxies, and module identities.

At the time of writing, the mgmt (2) subtree has only one subtree, mib-2 (1), defined. Figure B.4 shows the subtrees for mib-2. For example, according to the tree any object definition regarding an IP module would contain the object identifier 1.3.6.1.2.1.4.

Object definitions are generally packaged into **information modules**. Three types of information modules are defined using the SMI:

- MIB modules, which serve to group definitions of interrelated objects.
- Compliance statements for MIB modules. These define a set of requirements that managed nodes must meet with respect to one or more MIB modules.
- Capability statements for agent implementations. These specify the degree to which a managed node is able to implement objects that are defined in a MIB module. Capability statements are often provided by vendors with regard to a particular product and how well it can implement particular MIB modules.

The names of all standard information modules must be unique. Information modules that are developed for a particular company, known as enterprise information modules, must have names that are unlikely to match a standard name.

B.4 MANAGEMENT INFORMATION BASE

The **Management Information Base (MIB)** is a virtual database used to define the functional and operational aspects of network devices. The database should contain an object for each functional aspect of a device that needs to be managed. These objects are usually grouped into different information modules. The information provided by the MIB represents the common view and structure of management capabilities that are shared between the management station and device's agent.

Each definition of a particular object contains the following information about the object: its name, the data type, a human-readable description, the type of access (read/write), and an object identifier. For example, the ip (4) subtree in Figure B.2 refers to an ip group that contains a number of object definitions pertaining to common ip

objects. One such object is:

```

ipInHdrErrors OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The number of input datagrams discarded due to
        errors in their IP headers, including bad
        checksums, version number mismatch, other format
        errors, time-to-live exceeded, errors discovered in
        processing their IP options, etc."
    ::= { ip4 }

```

We see that a manager can use this managed information to obtain statistics of the number of IP packets that are discarded because of various errors. Many more objects have been defined in the mib-2 subtree (See RFC 1213 and its recent updates).

B.5 REMOTE NETWORK MONITORING

An additional set of modules, known as Remote Network Monitoring (RMON), was developed in 1995. These are considered to be not only an extension of the mib-2 but also an improvement. In SNMP, managed information that is to be used to monitor a device must be collected by polling. Even if trap-based polling is used, a certain amount of overhead is associated with obtaining the information. RMON uses a technique called *remote management* to obtain monitoring data. In this approach a network monitor (often called a *probe*) collects the data from the device. The probe may stand alone or be embedded within the managed device. Management applications communicate with an RMON agent in the probe by using SNMP; they do not communicate directly with the device itself. This separation from the device makes it easier to share information among multiple management stations. Furthermore, if the management application loses its connection to the RMON agent, the application can usually retrieve the data later, as the RMON agent will continue collecting data (assuming it is able to) even in the absence of a connection to the management application. Because a probe has considerable resources, it can also store various historical statistical information that can later be played back by the network management station. Several switch manufacturers incorporate RMON software in their switches so as to facilitate network management of distributed LANs.

RMON also provides for a higher level of standardization of the information collected. The data collected by mib-2, while standard, is relatively raw and is generally in the form of counters. RMON turns the raw data into a form more suitable for management purposes.

RMON is included as a subtree of mib-2 (rmon (16)), as shown in Figure B.4. Its objects are divided into 10 subtrees based on their function. RMON focuses on network management at layer 2 (data link). An extension of RMON, RMON-2, was proposed to provide network management layer 3 (network) and higher but especially for network layer traffic.

FURTHER READING

2. International Standard 8824. Information processing systems—Open Systems Interconnection—“Specification of Abstract Syntax Notation One (ASN.1),” International Organization for Standardization, December 1987.
- Leinwand, A. and K. Fang, *Network Management: A Practical Perspective*, Addison-Wesley, Reading, Massachusetts, 1993.
- Perkins, D. T., *RMON: Remote Monitoring of SNMP-Managed LANs*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1999.
- Rose, M., *The Simple Book: An Introduction to Networking Management*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1996. Provides an excellent discussion of SNMP.
- Stallings, W., *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards*, Addison-Wesley, Reading, Massachusetts, 1993.
- RFC 1155, M. Rose and K. McCloghrie, “Structure and Identification of Management Information for TCP/IP-Based Internets,” May 1990.
- RFC 1157, J. Case, M. Fedor, M. Schoffstall, and J. Davin, “A Simple Network Management Protocol (SNMP),” May 1990.
- RFC 1213, K. McCloghrie and M. Rose, “Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II,” March 1991.
- RFC 1905, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, “Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2),” January 1996.
- RFC 1907, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, “Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2),” January 1996.
- RFC 2578, K. McCloghrie, D. Perkins, J. Shoenfelder, eds., “Structure of Management Information for Version 2” (authors of previous version: J. Case, K. McCloghrie, M. Rose, and S. Waldbusser), April 1999.
- RFC 2271, D. Harrington, R. Presuhn, and B. Wijnen, “An Architecture for Describing SNMP Management Frameworks,” January 1998. This RFC documents SNMPv3 architecture.
- RFC 2274, U. Blumenthal and B. Wijnen, “User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3),” January 1998.

See our website for additional references available through the internet.

An intriguing possibility is the deployment of community fiber access networks as an essential infrastructure, like the water supply. Such systems would enable very high-speed, Ethernet-based access from the residence. Such deployment is currently only within the reach of well-to-do communities and underscores the growing gap between the haves and the have-nots.

The purpose of networks is to support applications, so it is not surprising that the action will be at the edge of the network where the clients and the servers are located. Applications are being deployed that include multimedia such as various forms of video streaming; real-time services such as Internet telephony and associated mobility services; and, of course, web-based services and various forms of electronic commerce. We saw in Chapter 10 that the control of network-based services depends on a variety of servers. Policy servers will be required to control the network as well as the services provided over the network. In addition to name servers, location servers will be required to enable communications with mobile users. We saw in Chapter 11 that security servers will be required to manage keys and associated certificates. Bandwidth brokers will be required to control access to QoS services as well as to manage bandwidth resources within a domain. Access servers and firewalls will be needed to control access to private networks and other important resources.

New generations of wireless networking technologies are pushing towards a convergence of telephone networks and the Internet. A new generation of digital cellular phones is introducing a variety of new communication services. Texting or short message service has grown dramatically and is of particular interest because it combines the message capabilities of cellular phones with the Internet. The 802.11 wireless LAN is also promoting change in various fronts. 802.11 provides high speed Internet access to wireless devices such as laptops and PDAs in public hot spots such as airport lounges and restaurants. 802.11 when combined with DSL or cable modem access can potentially completely alter how communication takes place in the home and small business. A processor built into the DSL or cable modem can provide various server capabilities and so coordinate communication between various devices, for example, cordless phones, television, DVDs, MP3 players, game stations, PDAs, laptops, personal computer, and various types of appliances.

Network security is becoming a preeminent issue in network evolution. The breaking of the 802.11 security scheme shows that security remains a key unaddressed issue in wireless access. Recent successful attempts at disrupting the Internet through denial-of-service attacks via various worms demonstrate that IP networks have not yet attained the desired level of robustness and reliability as expected in public-switched telephone networks. The selection of encryption schemes that assure users of some degree of privacy while allowing governments interception capability is a difficult balance to strike in countries that value freedom of expression.

This discussion of network trends is not intended to be exhaustive. For example, we have not touched upon the important issues of business competition, regulation, and public policy. The recent surge and collapse in the telecom service and equipment industry demonstrates once again that nontechnical issues ultimately determine the mix of technology that is actually deployed. On the other hand, the Napster and peer-to-peer networking phenomena also demonstrate how new technologies can undermine entrenched business models, for example, the music and video business. It should be evident to the student that much of the story of networks is yet to unfold.

APPENDIX A

Delay and Loss Performance

A key feature of communication networks is the sharing of resources such as transmission bandwidth, storage, and processing capacity. Because the demand for these resources is unscheduled, the situation can arise where resources are not available when a user places a request. This situation typically leads to a delay or loss in service. In this appendix we develop some simple but fundamental models to quantify the delay and loss performance. The appendix is organized as follows:

1. *Little's formula* relates the average occupancy in the system to the average time spent in the system. This formula is extremely powerful in obtaining average delay performance of complex systems.
2. A *basic queueing model* for a multiplexer allows us to account for arrival rate, message length, transmission capacity, buffer size, and performance measures such as delay and loss.
3. The *M/M/1* model provides a simple, basic multiplexer model that allows us to explore trade-offs among the essential system parameters.
4. The *M/G/1* model provides a more precise description of service times and message lengths.
5. The *Erlang B blocking formula* quantifies blocking performance in loss systems.

A.1 DELAY ANALYSIS AND LITTLE'S FORMULA

Figure A.1 shows a basic model for a delay/loss system. Customers arrive to the system according to some arrival pattern. These customers can be connection requests, individual messages, packets, or cells. The system can be an individual transmission line, a multiplexer, a switch, or even an entire network. The customer spends some time T in the system. After this time the customer departs the system. It is possible

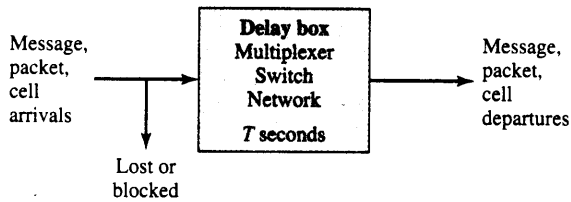


FIGURE A.1 Network delay analysis.

that under certain conditions the system is in a blocking state, for example, due to lack of resources. Customers that arrive at the system when it is in this state are blocked or lost. We are interested in the following performance measures:

- Time spent in the system: T .
- Number of customers in the system: $N(t)$.
- Fraction of arriving customers that are lost or blocked: P_b .
- Average number of customers/second that pass through the system: throughput.

Customers generally arrive at the system in a random manner, and the time that they spend in the system is also random. In this section we use elementary probability to assess the preceding performance measures.

A.1.1 Arrival Rates and Traffic Load Definitions

We begin by introducing several key system variables and some of their averages. Let $A(t)$ be the number of arrivals at the system in the interval from time 0 to time t . Let $B(t)$ be the number of blocked customers and let $D(t)$ be the number of customer departures in the same interval. The number of customers in the system at time t is then given by

$$N(t) = A(t) - D(t) - B(t) \quad (\text{A.1})$$

because the number that have entered the system up to time t is $A(t) - B(t)$ and because $D(t)$ of these customers have departed by time t . Note that we are assuming that the system was empty at $t = 0$. The long-term **arrival rate** at the system is given by

$$\lambda = \lim_{t \rightarrow \infty} \frac{A(t)}{t} \text{ customers/second} \quad (\text{A.2})$$

The **throughput** of the system is equal to the long-term departure rate, which is given by

$$\text{throughput} = \lim_{t \rightarrow \infty} \frac{D(t)}{t} \text{ customers/second} \quad (\text{A.3})$$

The **average number in the system** is given by

$$E[N] = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t N(t') dt' \text{ customers} \quad (\text{A.4})$$

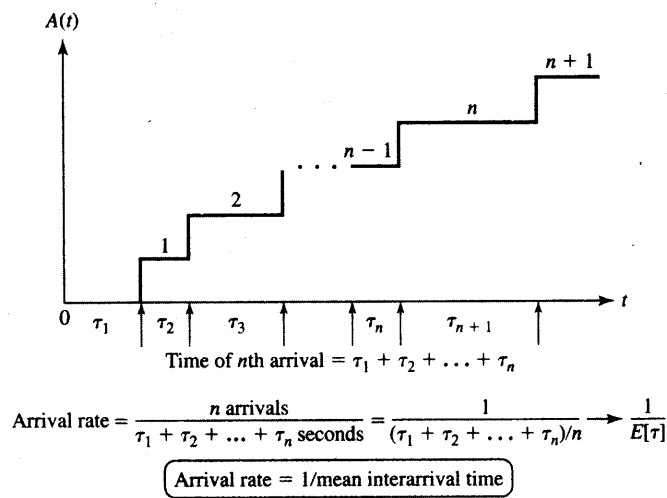


FIGURE A.2 Arrivals at a system as a sample function.

The fraction of blocked customers is then

$$P_b = \lim_{t \rightarrow \infty} \frac{B(t)}{A(t)} \tag{A.5}$$

Figure A.2 shows a typical sample function $A(t)$, the number of arrivals at the system. We assume that we begin counting customers at time $t = 0$. The first customer arrives at time τ_1 , and so $A(t)$ goes from 0 to 1 at this time instant. The second arrival is τ_2 seconds later. Similarly the n th customer arrival is at time $\tau_1 + \tau_2 + \dots + \tau_n$, where τ_i is the time between the arrival of the $i - 1$ and the i th customer. The arrival rate up to the time when the n th customer arrives is then given by $n/(\tau_1 + \tau_2 + \dots + \tau_n)$ customers/second. Therefore, the long-term arrival rate is given by

$$\lambda = \lim_{n \rightarrow \infty} \frac{n}{\tau_1 + \tau_2 + \dots + \tau_n} = \lim_{n \rightarrow \infty} \frac{1}{(\tau_1 + \tau_2 + \dots + \tau_n)/n} = \frac{1}{E[\tau]} \tag{A.6}$$

In Equation A.6 we assume that all of the interarrival times are statistically independent and have the same probability distribution and that their average or expected value is given by $E[\tau]$. Thus the average arrival rate is given by the reciprocal of the average interarrival time.

A.1.2 Little's Formula

Next we will develop **Little's formula**, which relates the average time spent in the system $E[T]$ to the arrival rate λ and the average number of customers in the system $E[N]$ by the following formula:

$$E[N] = \lambda E[T] \tag{A.7}$$

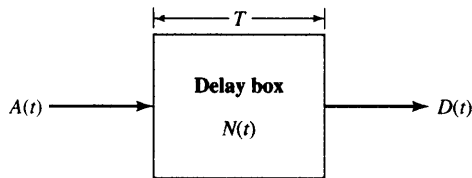


FIGURE A.3 Little's formula.

We will assume, as shown in Figure A.3, that the system does not block any customers. The number in the system $N(t)$ varies according to $A(t) - D(t)$.

Suppose we plot $A(t)$ and $D(t)$ in the same graph as shown in Figure A.4. $A(t)$ increases by 1 each time a customer arrives, and $D(t)$ increases by 1 each time a customer departs. The number of customers in the system $N(t)$ is given by the difference between $A(t)$ and $D(t)$. The number of departures can never be greater than the number of arrivals, and so $D(t)$ lags behind $A(t)$ as shown in the figure. Assume that customers are served in first-in, first-out (FIFO) fashion. Then the time T_1 spent by the first customer in the system is the time that elapses between the instant when $A(t)$ goes from 0 to 1 to the instant when $D(t)$ goes from 0 to 1. Note that T_1 is also the area of the rectangle defined by these two time instants in the figure. A similar relationship holds for all subsequent times T_2, T_3, \dots .

Consider a time instant t_0 where $D(t)$ has caught up with $A(t)$; that is, $N(t_0) - A(t_0) - D(t_0) = 0$. Note that the area between $A(t)$ and $D(t)$ is given by the sum of the times T_0 spent in the system by the first $A(t_0)$ customers. The time average of the number of customers in the system up to time t_0 is then

$$\frac{1}{t_0} \int_0^{t_0} N(t') dt' = \frac{1}{t_0} \sum_{j=1}^{A(t_0)} T_j \tag{A.8}$$

If we multiply and divide the preceding expression by $A(t_0)$, we obtain

$$\frac{1}{t_0} \int_0^{t_0} N(t') dt' = \frac{A(t_0)}{t_0} \left\{ \frac{1}{A(t_0)} \sum_{j=1}^{A(t_0)} T_j \right\} \tag{A.9}$$

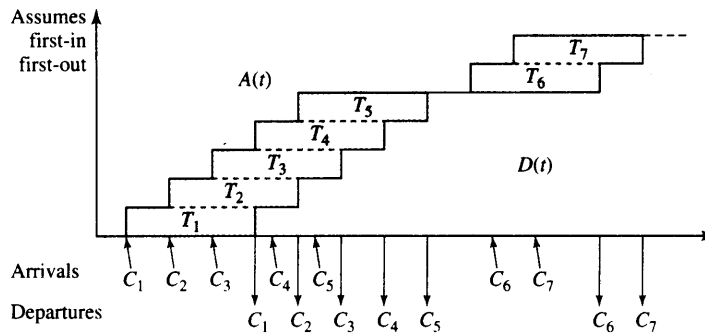


FIGURE A.4 Arrivals and departures in a FIFO system.

Equation A.9 states that, up to time t_0 , the average number of customers in the system is given by the product of the average arrival rate $A(t_0)/t_0$ and the arithmetic average of the times spent in the system by the first $A(t_0)$ customers. Little's formula as given by Equation A.7 then follows if we assume that

$$E[T] = \lim_{A(t_0) \rightarrow \infty} \left\{ \frac{1}{A(t_0)} \sum_{j=1}^{A(t_0)} T_j \right\} \quad (\text{A.10})$$

It can be shown that Little's formula is valid even if customers are not served in order of arrival [Bertsekas 1987].

Now consider a system in which customers can be blocked. The above derivation then applies if we replace $A(t)$ by $A(t) - B(t)$, the actual number of customers who enter the system. The actual arrival rate into a system with blocking is $\lambda(1 - P_b)$, since P_b is the fraction of arrivals that are blocked. It then follows that Little's formula for a system with blocking is

$$E[N] = \lambda(1 - P_b)E[T] \quad (\text{A.11})$$

In the preceding derivation we did not specify what constitutes a "system," so Little's formula can be applied in many different situations. Thus we can apply Little's formula to an individual transmission line, to a multiplexer, to a switch, or even to a network.

We now show the power of Little's formula by finding the average delay that is experienced by a packet in traversing a packet-switching network. Figure A.5 shows an entire packet-switching network that consists of interconnected packet switches. We assume that when a packet arrives at a packet switch the packet is routed instantaneously

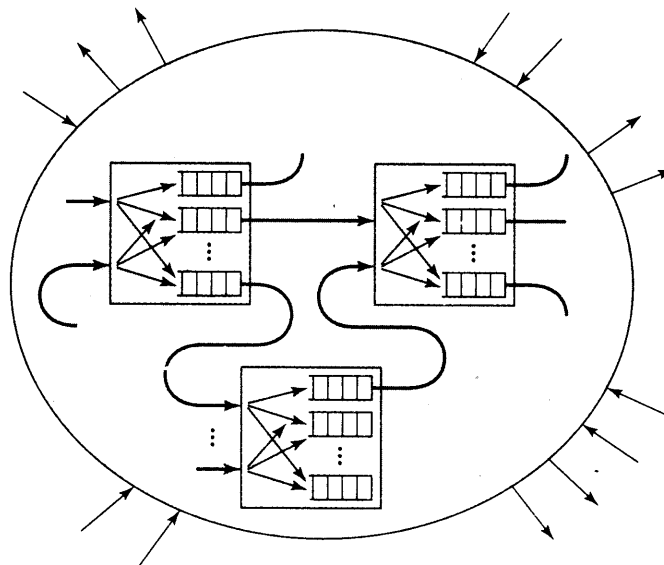


FIGURE A.5 Packet-switching network delay.

and placed in a multiplexer to await transmission on an outgoing line. Thus each packet switch can be viewed as consisting of a set of multiplexers. We begin by applying Little's formula to the network as a whole. Let N_{net} be the total number of packets in the network, let T_{net} be the time spent by the packet in the network, and let λ_{net} be the total packet arrival rate to the network, Little's formula then states that

$$E[N_{net}] = \lambda_{net} E[T_{net}] \quad (\text{A.12})$$

This formula implies that the average delay experienced by packets in traversing the network is

$$E[T_{net}] = E[N_{net}]/\lambda_{net} \quad (\text{A.13})$$

We can refine Equation A.13 by applying Little's formula to each individual multiplexer. For the m th multiplexer Little's formula gives

$$E[N_m] = \lambda_m E[T_m] \quad (\text{A.14})$$

where λ_m is the packet arrival rate at the multiplexer and $E[T_m]$ is the average time spent by a packet in the multiplexer. The total number of packets in the network N_{net} is equal to the sum of the packets in all the multiplexers:

$$E[N_{net}] = \sum_m E[N_m] = \sum_m \lambda_m E[T_m] \quad (\text{A.15})$$

By combining the preceding three equations, we obtain an expression for the total delay experienced by a packet in traversing the entire network:

$$E[T_{net}] = E[N_{net}]/\lambda_{net} = \frac{1}{\lambda_{net}} \sum_m \lambda_m E[T_m] \quad (\text{A.16})$$

Thus the average network delay depends on the overall arrival rates in the network, the arrival rate to individual multiplexers, and the delay in each multiplexer. The arrival rate at each multiplexer is determined by the routing algorithm. The delay in a multiplexer depends on the arrival rate and on the rate at which the associated transmission line can transmit packets. Thus the preceding formula succinctly incorporates the effect of routing as well as the effect of the capacities of the transmission lines in the network. For this reason the preceding expression is used in the design and management of packet-switching networks. To obtain $E[T_m]$, it is necessary to analyze the delay performance of each multiplexer. This is our next topic.

A.2 BASIC QUEUEING MODELS

The pioneering work by Erlang on the traffic engineering of telephone systems led to the development of several fundamental models for the analysis of resource-sharing systems. In a typical application customers demand resources at random times and

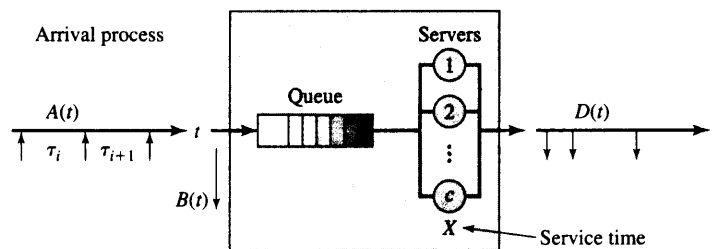


FIGURE A.6 Queueing model.

use the resources for variable durations. When all the resources are in use, arriving customers form a line or “queue” to wait for resources to become available. *Queueing theory* deals with the analysis of these types of systems.

A.2.1 Arrival Processes

Figure A.6 shows the basic elements of a queueing system. Customers arrive at the system with interarrival times $\tau_1, \tau_2, \dots, \tau_n$. We will assume that the interarrival times are independent random variables with the same distribution. The results for the arrival process developed in Figure A.2 then hold. In particular, the *arrival rate to the system* is given by

$$\lambda = \frac{1}{E[\tau]} \text{ customers/second} \quad (\text{A.17})$$

Several special cases of arrival processes are of interest. We say that arrivals are *deterministic* when the interarrival times are all equal to the same constant value. We say that the arrival times are *exponential* if the interarrival times are exponential random variables with mean $E[\tau] = 1/\lambda$:

$$P[\tau > t] = e^{-t/E[\tau]} = e^{-\lambda t} \quad \text{for } t > 0 \quad (\text{A.18})$$

The case of exponential interarrival times is of particular interest because it leads to tractable analytical results. It can be shown that when the interarrival times are exponential, then the number of arrivals $A(t)$ in an interval of length t is given by a Poisson random variable with mean $E[A(t)] = \lambda t$:

$$P[A(t) = k] = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad \text{for } k = 0, 1, \dots \quad (\text{A.19})$$

For this reason, the case of exponential interarrival times is also called the *Poisson arrival process*.

A.2.2 Service Times

Resources are denoted by “servers” because their function is to serve customer requests. The time required to service a customer is called the **service time** and is denoted by X .

In our discussion the server is typically a transmission line and the service time can be the time required to transmit a message or the duration of a telephone call. The maximum rate at which a server can process customers is attained when the server is continuously busy. When this is the case, the average time between customer departures is equal to the average service time. The processing capacity of a single server is given by the maximum throughput or departure rate. From the discussion leading to the arrival rate formula, clearly the processing capacity is given by

$$\mu = \frac{1}{E[X]} \text{ customers/second} \quad (\text{A.20})$$

The processing capacity μ can be likened to the maximum flow that can be sustained over a pipe. The number of servers c in a queueing system can be greater than one. The total processing capacity of a queueing system is then given by $c\mu$ customers/second.

An ideal queueing system is one where customers arrive at equal intervals and in which they require a constant service time. As long as the service time is less than the interarrival time, each customer arrives at an available server and there is no waiting time. In general, however, the interarrival time and the service times are random. The combination of a long service time followed by a short interarrival time can then lead to a situation in which the server is not available for an arriving customer. For this reason, in many applications a queue is provided so that a customer can wait for an available server, as shown in Figure A.6. When a server becomes available the next customer to receive service is selected according to the service discipline. Possible service disciplines are FIFO; last-in, first-out (LIFO); service according to priority class; and random order of service. We usually assume FIFO service disciplines.

The *maximum* number of customers allowed in a queueing system is denoted by K . Note that K includes both the customers in queue and those in service. We denote the total number of customers in the system by $N(t)$, the number in queue by $N_q(t)$, and the number in service by $N_s(t)$. When the system is full, that is, $N(t) = K$, then new customers arrivals are blocked or lost.

A.2.3 Queueing System Classification

Queueing systems are classified by a notation that specifies the following characteristics:

- Customer arrival pattern.
- Service time distribution.
- Number of servers.
- Maximum number in the system.

For example, in Figure A.7 the queueing system $M/M/1/K$ corresponds to a queueing system in which the interarrival times are exponentially distributed (M)¹; the service times are exponentially distributed (M); there is a single server (1); and at most K customers are allowed in the system. The $M/M/1/K$ model was used to illustrate the

¹The notation M is used for the exponential distribution because it leads to a Markov process model.

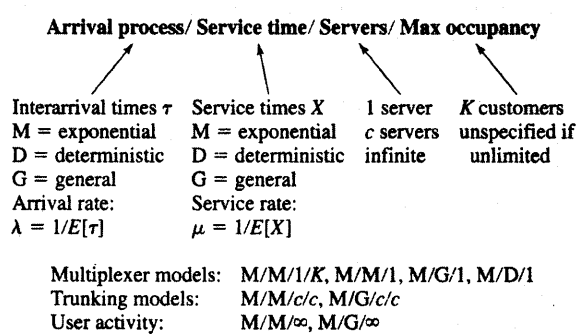


FIGURE A.7 Queuing model classification.

typical delay and loss performance of a data multiplexer in Chapter 5. If there is no maximum limit in the number of customers allowed in the system, the parameter K is left unspecified. Thus the M/M/1 system is identical to the above system except that it has no maximum limit on the number of customers allowed in the system.

The M/G/1 is another example of a queuing system where the arrivals are exponential, the service times have a *general* distribution, there is a single server, and there is no limit on the customers allowed in the system. Similarly, the M/D/1 system has constant, that is, *deterministic*, service times.

Figure A.8 shows the parameters that are used in analyzing a queuing system. The total time that a customer spends in the system is denoted by T , which consists of the time spent waiting in queue W plus the time spent in service X . When a system has blocking, P_b denotes the fraction of customers that are blocked. Therefore, the actual arrival rate into the system is given by $\lambda(1 - P_b)$. This value is the arrival rate that should be used when applying Little's formula. Thus the average number in the system and the average delay in the system are related by

$$E[N] = \lambda(1 - P_b)E[T] \tag{A.21}$$

If we apply Little's formula where the "system" is just the queue, then the average number of customers in queue and the average waiting time are related by

$$E[N_q] = \lambda(1 - P_b)E[W] \tag{A.22}$$

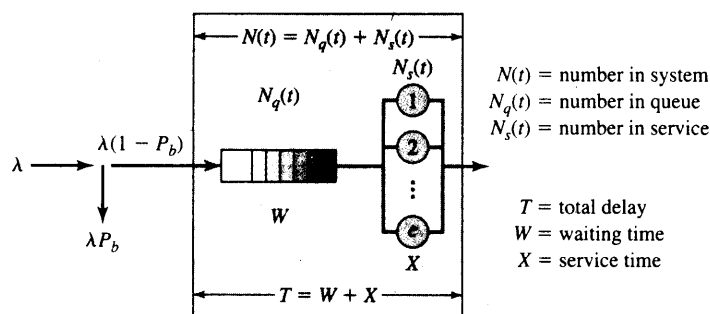


FIGURE A.8 Queuing system variables.

Finally, if the system is defined as a set of servers, then the average number of customers in service and the average service time are related by

$$E[N_s] = \lambda(1 - P_b)E[X] \tag{A.23}$$

The preceding three equations are very useful in relating average occupancy and average delay performance. Typically it is relatively simple to obtain the averages associated with occupancies, that is, $N(t)$, $N_q(t)$ and $N_s(t)$.

Finally, we revisit some of the terms introduced earlier in the appendix. The *traffic load* or offered load is the rate at which “work” arrives at the system:

$$\begin{aligned} a &= \lambda \text{ customers/second} \times E[X] \text{ seconds/customer} \\ &= \lambda/\mu \text{ Erlangs} \end{aligned} \tag{A.24}$$

The *carried load* is the average rate at which the system does work. It is given by the product of the average service time per customer, $E[X]$, and the actual rate at which customers enter the system, $\lambda(1 - P_b)$. Thus we see that the carried load is given by $a(1 - P_b)$.

The *utilization* ρ is defined as the average fraction of servers that are in use:

$$\rho = \frac{E[N_s]}{c} = \frac{\lambda}{c\mu}(1 - P_b) \tag{A.25}$$

Note that when the system has a single server, then the utilization ρ is also equal to the proportion of time that the server is in use.

A.3 M/M/1: A BASIC MULTIPLEXER MODEL

In this section we develop the M/M/1/K queueing system, shown in Figure A.9, as a basic model for a multiplexer. The interarrival times τ in this system have mean $E[\tau] = 1/\lambda$ as an exponential distribution. Let $A(t)$ be the number of arrivals in the interval 0 to t ; then as indicated above $A(t)$ has a Poisson distribution.

The average packet length is $E[L]$ bits per packet, and the transmission line has a speed of R bits/second. So the average packet transmission time is $E[X] = E[L]/R$ seconds. This transmission line is modeled by a single server that can process packets at a maximum rate of $\mu = R/E[L]$ packets/second. We assume that the packet transmission time X has an exponential distribution:

$$P[X > t] = e^{-t/E[X]} = e^{-\mu t} \text{ for } t > 0 \tag{A.26}$$

We also assume that the interarrival times and packet lengths are independent of each other. We will first assume that at most K packets are allowed in the system. We later consider the case where K is infinite.

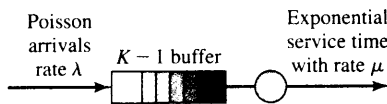


FIGURE A.9 M/M/1/K queue.

In terms of long-term flows, packets arrive at this system at a rate of λ packets/second, and the maximum rate at which packets can depart is μ packets/second. If $\lambda > \mu$, then the system will necessarily lose packets because the system is incapable of handling the arrival rate λ . If $\lambda < \mu$, then on the average the system can handle the rate λ , but it will occasionally delay and/or lose packets because of temporary surges in arrivals or long consecutive service times. We will now develop a model that allows us to quantify these effects.

Consider what events can happen in the next Δt seconds. In terms of arrivals there can be 0, 1, or > 1 arrivals. Similarly there can be 0, 1, or > 1 departures. It can be shown that if the interarrival times are exponential, then

$$P[1 \text{ arrival in } \Delta t] = \lambda \Delta t + o(\Delta t) \quad (\text{A.27})$$

where $o(\Delta t)$ denotes terms that are negligible relative to Δt , as $\Delta t \rightarrow 0$.² Thus the probability of a single arrival is proportional to λ . Similarly, it can also be shown that probability of no arrivals in Δt seconds is given by

$$P[0 \text{ arrival in } \Delta t] = 1 - \lambda \Delta t + o(\Delta t) \quad (\text{A.28})$$

The preceding two equations imply that only two events are possible as Δt becomes very small: one arrival or no arrival. Since the service times also have an exponential distribution, it can be shown that a customer in service will depart in the next Δt seconds with probability

$$P[1 \text{ departure in } \Delta t] = \mu \Delta t + o(\Delta t) \quad (\text{A.29})$$

and that the probability that the customer will continue its service after an additional Δt seconds is

$$P[0 \text{ departure in } \Delta t] = 1 - \mu \Delta t + o(\Delta t) \quad (\text{A.30})$$

A.3.1 M/M/1 Steady State Probabilities and the Notion of Stability

We can determine the probability of changes in the number of customers in the system by considering the various possible combinations of arrivals and departures:

$$\begin{aligned} P[0 \text{ arrival \& 0 departure in } \Delta t] &= \{1 - \mu \Delta t + o(\Delta t)\} \{1 - \lambda \Delta t + o(\Delta t)\} \\ &= 1 - (\lambda + \mu) \Delta t + o(\Delta t) \end{aligned} \quad (\text{A.31})$$

Equation A.31 gives the probability that the number in the system is still $n > 0$ after Δt seconds.

$$\begin{aligned} P[1 \text{ arrival \& 0 departure in } \Delta t] &= \{\lambda \Delta t + o(\Delta t)\} \{1 - \mu \Delta t + o(\Delta t)\} \\ &= \lambda \Delta t + o(\Delta t) \end{aligned} \quad (\text{A.32})$$

²In particular, a function $g(x)$ is $o(x)$ if $g(x)/x \rightarrow 0$ as $x \rightarrow 0$; that is, $g(x)$ goes to 0 faster than x does.